Received: October 3, 2024. Revised: November 25, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Multi-Cluster DBSCAN for Analysing Tourism Data

Edi Faizal[1,2] **Sri Hartati**[1]* **Aina Musdholifah**[1]

[1]*Department of Computer Science and Electronics, Faculty of Mathematics and Computer Sciences, Universitas Gadjah Mada, Yogyakarta, Indonesia*
[2]*Software Engineering, Faculty of Information Technology, Universitas Teknologi Digital Indonesia, Yogyakarta, Indonesia*
*Corresponding author's Email: shartati@ugm.ac.id

Abstract: This paper presents a novel multi-clustering approach using the optimized DBSCAN algorithm to analyze tourism data with diverse and complex attributes. Through feature engineering, we transformed raw data into more informative representations. By fine-tuning DBSCAN parameters and using five distance metrics, the optimal clustering configuration for 347 destinations was identified. The experimental results show that DBSCAN significantly outperformed K-Means and FCM, achieving perfect Silhouette Scores for categorical features, type, and popularity. In addition, DBSCAN demonstrated superior cluster separation and density, as reflected by lower DBI and higher CHI values. For geographic features, DBSCAN achieved the highest Silhouette Score (0.54940), surpassing K-Means (0.48374) and FCM (0.45724), despite challenges in clustering spatial data. DBSCAN also recorded the shortest computation time, highlighting its efficiency. This research underscores the importance of feature engineering and parameter tuning in gaining deeper insights and improving the clustering process for tourism data analysis.

Keywords: Clustering, Multi-clustering, Dbscan, Distance metrics, Tourism data.

1. Introduction

The global expansion of tourism in recent decades has generated a vast and complex dataset, encompassing a wide range of critical variables such as visit statistics, reviews, destination ratings, and geographic attributes. These elements are vital for understanding tourist behavior and destination dynamics, as rich data can provide deep insights into tourist preferences and behavioral trends. The rapid growth of the tourism sector has driven the need for the application of sophisticated and innovative analytical methods to identify meaningful patterns and trends within this data, making it increasingly crucial to optimize destination management strategies and develop policies that are responsive to market demands [1, 2].

However, despite the significant potential of this data, tourism data analysis often faces substantial challenges, particularly in clustering destinations based on visit patterns, demographic characteristics, and other attributes. To address these limitations, this study proposes a multi-clustering approach utilizing an optimized Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm, aiming to provide a more effective solution for identifying representative clusters in tourism data and enhancing the quality of analysis and understanding of destination dynamics [3–6].

Traditional clustering methods like K-Means and Fuzzy C-Means (FCM) are often inadequate for handling the complexities of tourism data, which frequently contains irregularly shaped clusters, varying densities, and noise [7]. DBSCAN, introduced by Ester et al. [3], was selected for its ability to identify arbitrarily shaped clusters and effectively manage noise, making it well-suited for the diverse nature of tourism data. Consequently, multi-clustering offers a more detailed and nuanced understanding of tourism data [8–10]. However, DBSCAN's performance is highly dependent on two key parameters: epsilon (*eps*), which defines the maximum distance between two points to be

considered part of the same cluster, and minimum points (*minPts*), which specifies the minimum number of points required to form a cluster [11]. To address this, the study optimizes DBSCAN parameters through a systematic grid search and applies five different distance metrics—Euclidean, Cosine, Cityblock, Minkowski, and Hamming—to enhance the algorithm's performance and clustering results.

The uniqueness of this study lies in the application of DBSCAN with various configurations and distance metrics, offering an adaptive multi-clustering framework. This approach allows for the identification of clusters with different shapes and densities, commonly found in tourism data, which is a mix of quantitative and qualitative features such as geographic data, visitor types, and destination ratings. Furthermore, by employing a multi-clustering approach, clustering results can be cross-validated to ensure robustness and reliability [12, 13]. With proper DBSCAN parameter adjustments, this method is capable of producing more accurate and meaningful clustering outcomes.

Ultimately, this study offers a more effective and relevant clustering approach for complex tourism data. The proposed multi-clustering approach contributes to theoretical knowledge while providing practical tools and insights that support better destination management and marketing strategies. By integrating various distance metrics and systematically optimizing DBSCAN parameters, this research delivers a robust and adaptive approach to clustering analysis, yielding deep insights into the dynamics and patterns within tourism data [14, 15].

A critical aspect of clustering analysis is balancing intra-cluster cohesion and inter-cluster separation, aiming to achieve meaningful and actionable clustering results. In this study, clustering quality is assessed using advanced evaluation metrics such as Silhouette Score (Sil_Score), Davies-Bouldin Index (DBI), and Calinski-Harabasz Index (CHI), which provide a comprehensive evaluation of the internal consistency of clusters and the degree of separation between them [16–21]. These metrics offer a robust framework for comparing the performance of various clustering configurations and ensuring optimal outcomes.

While clustering has been widely applied in tourism data analysis, this study introduces novelty through a multi-clustering framework that leverages an optimized DBSCAN algorithm with multiple distance metrics. This multi-configuration approach accurately captures the nuances of multidimensional tourism data, enabling the detection of irregularly shaped and variably dense clusters while addressing the limitations of traditional DBSCAN. The proposed method not only provides a powerful mechanism for analyzing complex tourism datasets but also offers valuable insights for theoretical advancement and practical applications in tourism management. The optimized DBSCAN method presented in this study advances state-of-the-art tourism data analysis by providing a flexible, robust, and scalable clustering framework. Through systematic parameter optimization and applying multiple distance metrics, this research introduces a novel approach to uncovering hidden patterns in tourism data, ultimately improving strategic decision-making for tourism stakeholders.

This paper is structured as follows: Section 2 reviews clustering algorithms in tourism data analysis. Section 3 outlines the methodology, including the multi-clustering approach and DBSCAN optimization with various distance metrics. Section 4 discusses experimental results, demonstrating the method's effectiveness across feature subsets. Finally, Section 5 concludes by highlighting how the method identified destination based clusters, improved recommendations, and addressed limitations of traditional clustering approaches to enhance complex tourism data management.

2. Related work

2.1 Clustering algorithms in data analysis

Clustering algorithms are essential techniques for grouping data based on similar attributes, helping to uncover patterns and structures within datasets [22, 23]. Widely used algorithms such as K-Means and FCM are popular for a variety of data analysis tasks. These methods are highly effective when the dataset exhibits clear cluster boundaries. However, when data is complex, noisy, and varies in density, these algorithms often struggle to provide meaningful clusters. This is particularly true in the tourism domain, where data typically has varying densities and complex structures [24].

Among the clustering techniques, DBSCAN introduced by Ester et al. [3], is a standout due to its ability to detect clusters of arbitrary shapes and handle noise. Unlike K-Means, which assumes clusters are spherical and of similar sizes, DBSCAN can adapt to varying densities, making it especially suitable for complex datasets such as those found in tourism data. Tourism data often exhibit heterogeneous features, such as varying tourist preferences, behaviors, and movements, making DBSCAN a compelling choice for clustering tourists

based on their activities, motivations, and preferences [20, 25].

Recent studies have demonstrated the application of clustering techniques to tourism data, highlighting their importance in destination profiling and market segmentation. For example, DBSCAN has been employed to cluster tourists based on their movement patterns, preferences, and behaviors [26]. These studies emphasize that clustering can aid in identifying distinct tourist segments, leading to more effective marketing strategies and personalized destination recommendations.

2.2 Parameter tuning and distance metrics

Effective parameter selection and the choice of distance metrics are critical to enhancing DBSCAN's performance, particularly when dealing with diverse, noisy, and complex datasets. One commonly used approach for parameter selection involves the k-distance graph, which helps identify optimal values for the key DBSCAN parameters, such as *eps* and *minPts* [27]. This method has proven useful in determining the appropriate values for these parameters, allowing DBSCAN to better adapt to the inherent complexities of tourism data.

Additionally, various optimization algorithms have been explored to improve DBSCAN's clustering outcomes. For example, integrating nearest neighbor search mechanisms has been shown to mitigate the over-identification of noise points and improve clustering accuracy [28]. These advancements contribute to more effective clustering, especially in datasets with significant noise or varying densities.

The selection of a suitable distance metric is equally important in clustering tasks. DBSCAN's performance can vary significantly depending on the distance metric used. Different metrics, can capture different aspects of the data's structure, and choosing the appropriate metric is crucial for obtaining meaningful clusters [29]. This study aims to evaluate various distance metrics, within the DBSCAN framework for tourism data analysis. The goal is to enhance clustering results by identifying the most suitable metric for the heterogeneous nature of tourism datasets.

2.3 Multi-Clustering approaches in tourism data

The multi-clustering approach, which combines multiple clustering techniques, is gaining popularity in tourism data analysis. This approach can provide deeper insights into tourist preferences and behaviors by clustering data based on multiple features or subsets of features [30]. For example, by using different clustering algorithms or feature sets, researchers can gain a more nuanced understanding of tourist segments based on destination preferences, motivations, satisfaction, and prior knowledge [7].

In tourism research, multi-clustering methods have been applied to group tourists based on a variety of attributes, such as their travel behavior, demographics, and past visitation patterns [31]. These methods are useful for high-dimensional data, as clustering different feature subsets can reveal hidden patterns. However, researchers should be aware of the limitations of the algorithms, especially when handling mixed data types [7].

This study implements a multi-clustering approach that focuses on DBSCAN, exploring different feature combinations and incorporating new validity indices to evaluate the clustering results. By leveraging this approach, the research seeks to contribute to the existing literature on tourism clustering by demonstrating how DBSCAN can effectively handle the challenges posed by tourism data, including its diversity and complexity.

2.4 State of the art

This research introduces a novel approach to tourism data analysis by applying the DBSCAN algorithm with optimized parameters and distance metrics. The study uses k-distance analysis for automatic parameter selection, ensuring the algorithm adapts to varying data densities and structures. Additionally, by testing different distance metrics, the research aims to improve clustering performance for heterogeneous tourism datasets, which often include a mix of categorical, numerical, and spatial data.

The integration of validity indices tailored to tourism data further enhances the clustering analysis, providing robust tools for evaluating clustering outcomes in the context of tourism data. These advancements make DBSCAN a more effective tool for tourism data analysis, offering insights into tourist behavior, segmentations, and destination profiling. By applying DBSCAN with optimized parameters, this research aims to enhance clustering accuracy and address the unique challenges posed by tourism datasets, such as density variation and the presence of noise.

3. Methodology

3.1 Research design

This study aims to apply a multi-clustering approach using the DBSCAN algorithm to analyze a

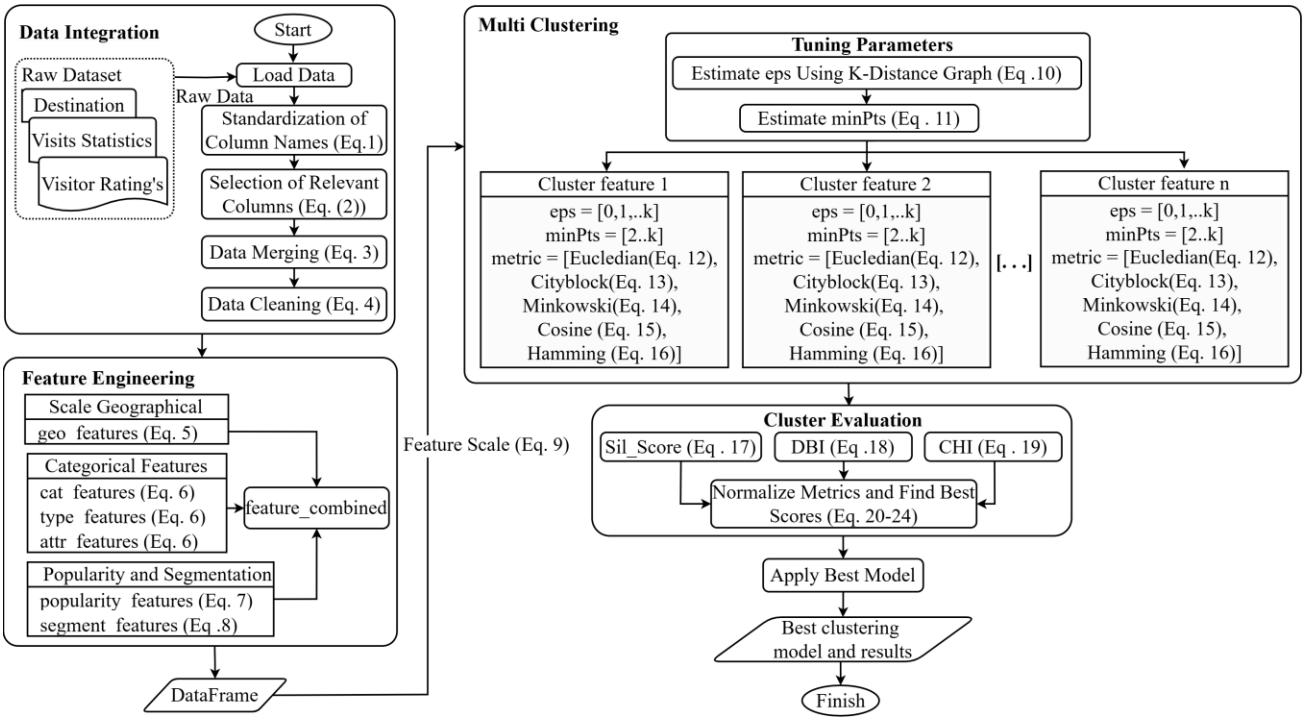

Figure. 1 The proposed Multi-Cluster DBSCAN for analysing tourism data

dataset of tourism destinations. The methodological process involves five key stages: data integration, eature engineering, parameter tuning, multi-clustering, and evaluation. The workflow of the proposed methodology is illustrated in Fig. 1. Throughout these stages, various equations are employed, with all symbols used in the equations detailed in Table 1.

3.2 Data integration

The data integration phase is pivotal, consolidating heterogeneous data sources into a unified and cohesive dataset, which serves as the foundation for subsequent analysis. This phase involves systematic steps of standardization, selection, merging, and cleansing.

The data integration phase plays a critical role in consolidating multiple heterogeneous data sources into a unified and cohesive dataset, which forms the foundation for subsequent analyses such as clustering. This phase includes systematic steps of standardization, selection, merging, and cleaning.

Standardization of column names: To ensure consistency and avoid case sensitivity across the dataset, all column names are converted to lowercase Eq. (1).

$$C_i = lowercase(C_i)\ \forall_i \in \{1,2,\dots,n\} \quad (1)$$

where C_i represents the i-th column name in the dataset, and n is the total number of columns. This step simplifies data processing and minimizes potential errors during feature extraction and transformation.

Selection of relevant columns: The dataset comprises multiple dataframes, including destination, statistic, and visitor rating, each containing diverse attributes of tourism destinations. Only the relevant columns $C \subseteq D_i$ are selected from the dataframe D_i, where C represents the key features required for clustering analysis Eq. (2).

$$C = \{lat, longt, category, rating, \dots\} \quad (2)$$

The destination dataframe consists of important columns, including geographical coordinates (lat, longt), category, types, and attractions. The statistic dataframe includes data on domestic and foreign visitors to each destination, while the visitor rating dataframe contains the ratings provided by visitors for the destinations they visited. These features are essential for understanding the structure and behavior of tourism destinations and form the basis for clustering analysis.

Data merging: The selected dataframes D_1 (destination), D_2 (statistic), and D_3 (visitor rating) are merged based on a unique key K (e.g., destno), ensuring that all relevant attributes from each source are integrated Eq. (3).

Table 1. Symbols and descriptions used in equations

Symbols	Description
C_i,C, n	C_i refers to the i-th column in the dataset, C is the set of relevant columns selected for analysis, and n is the total number of columns or data points (Eq. 1, 2).
Z, D_1, D_2, D_3, K	Z is the merged dataset created from dataframes D_1, D_2, D_3 using the unique key K (typically *'destno'*) as the merging key (Eq. 3).
X_{irr}, X_i, μ_i, σ_i, $geo\_feature_i$	X_{irr} are irrelevant or redundant columns removed from the dataset. X_i is the original value of a geographical feature, μ_i is its mean, σ_i is its standard deviation, and $geo\_feature_i$ is the normalized version of this feature (*latitude*, *longitude*) (Eq. 4, 5).
X_{ij}, c_j	X_{ij} is a binary indicator for the i-th observation and j-th category in one-hot encoding. c_j represents unique categories involved in the encoding (Eq. 6).
pop, *rating*, *ndom*, *nfor*, *Low*, *High*, *seg*	*pop* represents the customer satisfaction level categorized as *Poor*, *Good*, or *Excellent* based on the rating. *rating* is the satisfaction score, *ndom* and *nfor* represent the number of domestic and foreign visitors, respectively. *High* and *Low* refer to the visitor count categories, while *seg* indicates the visitor group: *domestic*, *foreign*, *allsegment*, or *nosegment*. (Eq. 7, 8).
x_i, $\bar{x}$, s, z_i	x_i is the original feature value before standardization, $\bar{x}$ is the mean of that feature, s is its standard deviation, and z_i is the standardized feature value (Eq. 9).
$d_{line}(x)$, p_1, $\bar{v}$	$d_{line}(x)$ is the *K-nearest neighbor* distance used to determine the parameter *eps* in DBSCAN. p_1 is the first point on the *K-nearest neighbor* distance plot, and $\bar{v}$ is the vector from the start to the end of the distance plot (Eq. 10).
d	d is the number of feature dimensions involved in clustering (Eq. 11).
δ	δ is the Kronecker delta function, which returns 1 if $x_i \neq y_i$ and 0 otherwise. It is commonly used in conjunction with Hamming distance (Eq. 16).
a_i, b_i, Sil_Score	a_i is the average distance from sample i to points in the same cluster, b_i is the average distance from sample i to points in the nearest neighboring cluster, and Sil_Score is the Silhouette Score used to measure clustering quality (Eq. 17).
S_i, S_j, d_{ij}, k, DBI	S_i and S_j are the average intra-cluster distances for clusters i and j. d_{ij} is the distance between the centroids of clusters i and j, k is the total number of clusters, and DBI is the Davies-Bouldin Index, which measures the similarity between clusters (Eq. 18).
B_k, W_k, CHI	B_k is the between-cluster dispersion matrix, W_k is the within-cluster dispersion matrix, and CHI is the Calinski-Harabasz Index, which calculates the ratio of between-cluster to within-cluster dispersion (Eq. 19).
N_S, N_D, N_C S_{min}, S_{max} DBI_{min}, DBI_{max}, CHI_{min}, CHI_{max}	These represent the normalized versions of the Sil_Score (N_S), DBI (N_D), and CHI (N_C), along with their minimum and maximum values (S_{min}, S_{max}, DBI_{min}, DBI_{max}, CHI_{min}, CHI_{max}). The normalized values are used to standardize the scores for comparison and evaluation of clustering performance (Eq. 20, 21, 22).
Avg_Score, *Best_Cluster*	*Avg_Score* is the average score across normalized metrics (Sil_Score, DBI, and CHI), and *Best_Cluster* is the cluster with the highest average score (Eq. 23, 24).

$$Z = D_1 \bowtie_K D_2 \bowtie_K D_3 \quad (3)$$

where Z represents the resulting unified dataset after the merging process.

Data cleaning: Post-merging, irrelevant or redundant columns X_{irr} are systematically removed to keep the dataset focused and manageable Eq. (4).

$$X = Z - X_{irr} \quad (4)$$

The cleaned dataset X is then prepared for feature engineering and clustering. This step ensures that the dataset remains efficient for further analysis.

3.3 Feature engineering

Feature engineering is a critical process that transforms raw data into meaningful input features, tailored specifically for DBSCAN clustering. This phase involves the application of advanced data transformation techniques, including normalization, one-hot encoding, and feature creation, designed to maximize the clustering algorithm's effectiveness.

***'geo_feature'*:** Geographical coordinates, represented by latitude (*lat*) and longitude (*longt*), are normalized to ensure comparability across these features, preventing any single attribute from disproportionately influencing the clustering results. This normalization process utilizes the

StandardScaler method and is mathematically expressed as follows Eq. (5).

$$geo\_feature_i = \frac{X_i - \mu_i}{\sigma_i},\ for\ i \in \{lat, longt\} \quad (5)$$

where $geo\_feature_i$ denotes the normalized geographical feature, X_i is the original value of the geographical feature, μ_i is the mean, and σ_i is the standard deviation of the respective geographical feature. The normalized values are subsequently stored in new columns labeled *'lat_scaled'* and *'longt_scaled'*.

Categorical features: Categorical attributes, such as ***'cat_features'***, ***'type_features'***, and ***'attr_features'***, often consist of comma-separated string values. To facilitate analysis, these string values are parsed into lists, followed by the application of one-hot encoding. This transformation is essential for converting categorical variables into binary indicator variables suitable for data processing algorithms.

The one-hot encoding process is critical, as categorical data must be represented in a numerical format to enable effective processing. Through one-hot encoding, each unique category of the variable is transformed into a binary column within a matrix, where each column represents a specific category. This formula effectively captures the logic of one-hot encoding, where each original category value is transformed into a binary vector, indicating the presence (1) or absence (0) of each category for each observation. This mathematical representation is expressed as follows Eq. (6):

$$X_{ij} = \begin{cases} 1, & if\ X_i = c_j\ (j = 1,2,\ldots,n) \\ 0, & otherwise \end{cases} \quad (6)$$

Where X_{ij} represents the binary indicator for the *i*-th observation and the *j*-th categorical, while c_j denotes the unique categories present within that feature. This one-hot encoding process is uniformly applied across all three features *'cat_features'*, *'type_features'*, and *'attr_features'* ensuring that all categorical variables are appropriately transformed into a binary format.

'popularity_features': The rating attribute, representing customer satisfaction, is transformed into categorical bins reflecting different levels of *popularity_features* (*pop*): *'Poor'*, *'Good'*, and *'Excellent'*. This transformation is achieved through binning, where rating values are divided into distinct categories, as shown in Eq. (7). One-hot encoding is then applied to convert these categories into binary columns with the prefix rating, ensuring precise representation during the clustering process.

$$pop_i = \begin{cases} \text{Poor}, & if\ rating_i \leq 2.9 \\ Good, & if\ 2.9 < rating_i \leq 4.0 \\ Excellent, & if\ rating_i > 4.0 \end{cases} \quad (7)$$

***'segment_features'*:** Visitor segmentation is crucial for understanding the different customer groups that a destination attracts. The *segment_features* (*seg*) are created based on the number of domestic (*ndom*) and foreign (*nfor*) visitors, categorized into *'Low'* and *'High'*. The binning for domestic visitors is defined as *binsdom*=[0, *avgdom* × 1, ∞], where *avgdom* is the mean number of domestic visitors. The segmentation process is identical for foreign visitors with corresponding bins: *binsfor*=[0, *avgfor* × 1, ∞].
An overall segmentation feature is derived based on logical rules evaluating the presence of domestic and foreign visitors, with categories such as domestic, foreign, allsegment, and nosegment Eq. (8). One-hot encoding is applied to these segments, generating binary columns prefixed with segment.

$$seg_i = \begin{cases} domestic, & if\ ndom_i = High\ \&\ nfor_i = Low \\ foreign, & if\ nfor_i = High\ \&\ ndom_i = Low \\ allsegment, & if\ ndom_i = High\ \&\ nfor_i = High \\ nosegment, & if\ ndom_i = Low\ \&\ nfor_i = Low \end{cases} \quad (8)$$

'feature_combined'*:** Following the engineering of individual features, they are aggregated into distinct subsets representing different aspects of the dataset. These subsets include ***'geo_features', ***'cat_features'***, ***'type_features'***, ***'attr_features'***, ***'popularity_features'*** and ***'segment_features'***, which encapsulates all engineered features for a comprehensive analysis.

The data integration process, including feature engineering and preparation for clustering analysis, which provides a step-by-step guide to achieving a clean and standardized dataset ready for analysis.

3.4 Multi-clustering

This section provides a detailed explanation of the multi-clustering process, which was conducted to analyze the dataset using various parameter combinations. DBSCAN was applied using five different distance metrics. Each combination was evaluated using three evaluation approaches, followed by fair normalization to determine the best score.

The first step in multi-clustering is to prepare the feature subsets and perform standardization. For each feature subset, relevant data is extracted and stored in *subset_df*. Standardization is performed using the StandardScaler to ensure all features are on the same

scale. Standardization transforms each feature x_i into z_i using Eq. (9).

$$z_i = \frac{x_i - \bar{x}}{s} \tag{9}$$

where x_i is the original feature value, $\bar{x}$ is the feature mean, and s is the feature standard deviation.

3.4.1 Parameter tuning

DBSCAN parameters were optimized using a grid search. The main parameters adjusted are *eps* and *minPts*. The *eps* parameter represents the maximum distance between two points to be considered in the same cluster, while *minPts* is the minimum number of points required to form a cluster.

The optimal *eps* value is determined by calculating the *K-neares neighbor* distance Eq. (10) for each point in the dataset and plotting it. The elbow point on the distance plot indicates the optimal value for *eps*.

$$d_{line}(x) = \frac{|(x - p_1)\cdot\bar{v}|}{|\bar{v}|} \tag{10}$$

where $\bar{v}$ is the vector from the start to the end of the plot, and p_1 is the first point on the plot.

The *minPts* estimate is calculated heuristically based on the number of data dimensions *d* Eq. (11).

$$minPts = \max(2 \times d, 4) \tag{11}$$

where *d* is the number of feature dimensions. The best parameters for *eps* and *minPts* are determined based on the analysis of the plot and heuristics.

3.4.2 Distance metrics

In this study, we employ five distinct distance metrics to compare clustering results. These metrics include Euclidean, Cityblock, Minkowski, Cosine, and Hamming distances.

Euclidean distance Eq. (12) is a commonly used metric to measure the straight-line distance between two points in Euclidean space.

$$d_{\text{Euclidean}}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{12}$$

where x and y are two points in n-dimensional space, and x_i and y_i are the i-th components of points x_i and y_i, respectively.

Cityblock distance Eq. (13), also known as Manhattan distance or *L1 Norm*, computes the distance between two points by summing the absolute differences of their coordinates. This metric is called Cityblock distance because it represents the total distance traveled along grid lines, similar to navigating a city grid layout.

$$d_{\text{Cityblock}}(x, y) = \sum_{i=1}^{n}|x_i - y_i| \tag{13}$$

where x and y are two points in n-dimensional space, x_i and y_i are the i-th components of points x and y, respectively.

Minkowski distance, as defined in Eq. (14), generalizes both Euclidean and Cityblock distances by introducing a parameter p. This metric allows for the measurement of distance between two points in a vector space, with its behavior varying according to the value of p. In this study, we employed p=3 to explore its effect on clustering results.

$$d_{\text{Minkowski}}(x, y) = \left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p} \tag{14}$$

where p is the distance parameter, x and y are two points in n-dimensional space, x_i and y_i are the i-th components of points x and y, respectively. If p=2, this reduces to Euclidean distance; if p=1, it reduces to Cityblock distance.

Cosine distance Eq. (15) measures the dissimilarity between two vectors based on the angle between them, rather than their magnitude. This distance metric is particularly useful in high-dimensional spaces where the direction of the vectors is more important than their length.

$$d_{\text{Cosine}}(x, y) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{15}$$

where x and y are vectors in n-dimensional space, x_i and y_i are the i-th components of vectors x and y, respectively.

Hamming distance Eq. (16) measures the proportion of differing components between two binary vectors. It counts the number of positions at which the corresponding bits are different.

$$d_{\text{Hamming}}(x, y) = \frac{1}{n}\sum_{i=1}^{n}\delta(x_i \neq y_i) \tag{16}$$

where δ is the Kronecker delta function, which is 1 if $x_i \neq y_i$ and 0 if $x_i = y_i$. x and y are binary vectors in n-dimensional space.

The choice of distance metric can significantly impact clustering outcomes. Experiments are conducted with different metrics to assess their effect on clustering performance.

3.4.3 Clustering evaluation

Clustering results are evaluated using several metrics. Sil_Score Eq. (17) measures the average distance between each sample and all other samples in its cluster compared to samples in other clusters.

$$\text{Sil\_Score}_i = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{17}$$

where a_i is the average distance between the sample and other points in the same cluster, and b_i is the average distance between the sample and points in the *nearest neighboring* cluster.

DBI Eq. (18) evaluates the average similarity ratio of each cluster with its most similar cluster.

$$\text{DBI} = \frac{1}{k}\sum_{i=1}^{k} \max_{j \neq i}\left(\frac{S_i + S_j}{d_{ij}}\right) \tag{18}$$

where k is the number of clusters, Si and S_j are the average distances within clusters i and j, and d_{ij} is the distance between the centroids of clusters i and j.

CHI Eq. (19) measures the ratio of between-cluster dispersion to within-cluster dispersion.

$$\text{CHI} = \frac{B_k}{W_k} \times \frac{n-k}{k-1} \tag{19}$$

where B_k is the between-cluster dispersion matrix, W_k is the within-cluster dispersion matrix, k is the number of clusters, and n is the number of samples.

For fair comparison, the evaluation metrics are first normalized. The normalization is performed for Sil_Score (N_S) Eq. (20), DBI (N_D) Eq. (21), and CHI (N_C) Eq. (22).

$$N_S = \frac{\text{Sil\_Score} - S_{min}}{S_{max} - S_{min}} \tag{20}$$

$$N_D = \frac{DBI_{max} - \text{DBI}}{DBI_{max} - DBI_{min}} \tag{21}$$

$$N_C = \frac{\text{CHI} - CHI_{min}}{CHI_{max} - CHI_{min}} \tag{22}$$

After normalization, the *Avg_Score* for each metric is calculated by Eq. (23). The best cluster Eq. (24) is the one with the highest average score.

$$Avg\_Score = \frac{N_S + N_D + N_C}{3} \tag{23}$$

$$Best\_Cluster = max\,(Avg\_Score) \tag{24}$$

The result of the best cluster is used to find the best parameter combination based on the highest average score. The best score for each feature subset and metric is identified and stored in a dataframe for further analysis. The best DBSCAN model is applied to the entire feature set using the identified optimal parameters. Through this process, a comprehensive multi-clustering analysis is conducted to find the optimal configuration of DBSCAN parameters that yields the best data clustering results.

4. Results and discussion

The dataset used in this study combines three data sources: destination data, visit statistics, and visitor ratings, encompassing 347 tourism destinations with features such as destination number, name, regency, category, attraction, type, coordinates, reviews, rating, and visitor count. Feature engineering was performed to prepare the data by scaling geographical features, one-hot encoding categorical attributes, and creating new features related to popularity and visitor segmentation, as previously described.

This chapter presents the clustering analysis results using various feature subsets, parameter combinations (*eps* and *minPts*), and distance metrics. The data, preprocessed through feature engineering, is evaluated based on Sil_Score, DBI, CHI, number of clusters, noise points, execution time, and normalized average scores. The experiments demonstrate how different feature subsets, parameters, and distance metrics influence clustering outcomes, revealing optimal configurations for uncovering patterns within the tourism dataset.

To validate our approach, we compared the optimized DBSCAN clustering method with alternative clustering techniques—K-Means and FCM—to assess clustering quality, performance, and computational efficiency across these methods. We further examined the impact of different feature engineering scenarios on clustering by applying all seven feature sets.

4.1 Experimental results

The clustering experiment results, presented in Table 2, are evaluated across seven distinct feature subsets: *geo_features*, *cat_features*, *type_features*, *attr_features*, *popularity_features*, *segment_features and feature_combined*. These subsets demonstrate performance variations influenced by their unique characteristics.

The table details the best performance results for each clustering method (DBSCAN, K-Means, and FCM), showcasing optimal parameters, Sil_Score, DBI, CHI, the number of clusters, noise points, and the time taken for each method to execute.

This comparison highlights how each clustering technique handles the different feature subsets and provides insights into the most effective configurations for clustering tourism data.

4.1.1 Geo_features

The *geo_features* subset, representing spatial data, produced mixed clustering outcomes across methods (see Table 2). DBSCAN with the Cityblock metric achieved a Sil_Score of 0.54940, forming two distinct clusters with minimal noise (4 data points). This score suggests some spatial separability, despite challenges from geographic variability.

In contrast, K-Means with 10 clusters had a lower Sil_Score (0.48374), struggling with spatial dispersion. FCM had the lowest score (0.45724), showing limitations with spatially diverse data. The variability in spatial data, such as distance differences between destinations, likely contributed to these results.

Table 2. Best performance results of each clustering method across different feature subsets

Subset	Method	Optimal Parameters	Sil_Score	DBI	CHI	N Clusters	N Noise	Time (seconds)	Avg Score
geo_ features	DBSCAN	*eps*=1, *minPts*=29, metric=Cityblock	0.54940	5.20E-01	1.37E+02	2	4	0.00740	0.56610
	K-Means	K=10	0.48374	6.98E-01	4.35E+02	10	n/a	0.01562	0.33819
	FCM	C=6	0.45724	7.02E-01	4.01E+02	6	n/a	0.01130	0.40255
cat_ features	DBSCAN	*eps*=3.1, *minPts*=20, metric=Euclidean	1	1.34E-08	8.19E+31	6	19	0.00110	0.99360
	K-Means	K=10	1	1.34E-08	8.19E+31	10	n/a	0.00001	1
	FCM	C=9	1	1.34E-08	8.19E+31	9	n/a	0.00912	1
type_ features	DBSCAN	*eps*=3.5, *minPts*=23, metric=Cityblock	0.99990	1.18E-08	4.08E+31	4	0	0.00030	0.82610
	K-Means	K=7	1	1.18E-08	4.08E+31	7	n/a	0.01201	0.83249
	FCM	C=7	1	1.18E-08	4.08E+31	7	n/a	0.00348	0.83249
attr_ features	DBSCAN	*eps*=2.2, *minPts*=2, metric=Cityblock	0.99710	1.07E-08	3.59E+31	5	1	0.00050	0.80570
	K-Means	K=7	0.99712	1.07E-08	3.59E+31	7	n/a	0.01568	0.81138
	FCM	C=7	0.99712	1.07E-08	3.59E+31	7	n/a	0.00001	0.81165
popularity_ features	DBSCAN	*eps*=2.5, *minPts*=3, metric=Euclidean	1	1.72E-08	7.14E+31	3	0	0.00100	0.95070
	K-Means	K=4	1	1.72E-08	7.14E+31	4	n/a	0.00110	0.95708
	FCM	C=7	1	1.72E-08	7.14E+31	7	n/a	0.00010	0.95708
segment_ features	DBSCAN	*eps*=0.3, *minPts*=2, metric=Cityblock	1	8.11E-09	7.48E+31	4	0	0.00100	0.96440
	K-Means	K=9	1	8.11E-09	7.48E+31	9	n/a	0.00929	0.97082
	FCM	C=5	1	8.11E-09	7.48E+31	5	n/a	0.01222	0.97082
feature_ combined	DBSCAN	*eps*=0.6, *minPts*=3, metric=Cosine	0.46700	3.92E-01	1.57E+01	2	1	0.01360	0.56890
	K-Means	K=2	0.46440	4.66E-01	1.59E+01	2	n/a	0.01156	0.36861
	FCM	C=10	0.04347	1.91E+00	2.01E+01	10	n/a	0.00110	0.11418

Table 3. Clustering performance of DBSCAN on *attr_features* Subset

eps	*minPts*	Metric	Sil_Score	DBI	CHI	N Clusters	N Noise	Time
2.2	2	Cityblock	0.99710	1.07E-08	3.59e+31	5	1	0.00050
0.6	3	Cosine	0.99710	1.07E-08	3.59e+31	5	1	0.00230
2.2	3	Euclidean	0.99710	1.07E-08	3.59e+31	5	1	0.00170
0.1	2	Hamming	0.99710	1.07E-08	3.59e+31	5	1	0.00900
0.9	3	Minkowski	0.99710	1.07E-08	3.59e+31	5	1	0.01560

DBSCAN's ability to identify dense regions and isolate noise made it more effective for *geo_features*, while K-Means and FCM struggled with uneven density. This confirms that density-based methods like DBSCAN are better for clustering spatial data with noise.

4.1.2 Cat_features

In the *cat_features* subset, which includes categorical attributes, all three methods demonstrated high clustering quality. DBSCAN, with optimized parameters (*eps*=3.1, *minPts*=20, metric=Euclidean), achieved a perfect Sil_Score of 1, identifying six clusters with minimal noise, indicating robust separability. K-Means and FCM similarly achieved strong clustering results, with 10 and 9 clusters respectively, highlighting this subset's suitability for clustering based on categorical features (see Table 2).

The strong clustering performance for *cat_features* across methods aligns with previous studies that emphasize the effectiveness of distance-based clustering for homogeneous, discrete data. The low DBI (close to 0) further validates the well-separated nature of these clusters, making *cat_feature* clustering suitable for identifying distinct destination types. DBSCAN's flexibility in handling minor noise at lower computational cost (0.00110 seconds) illustrates its advantage for *cat_feature* clustering in tourism datasets.

4.1.3 Type_features

For *type_features*, which describe destination types, all methods effectively clustered the data. DBSCAN and K-Means achieved near-perfect Sil_Scores (0.99990), forming four clusters without noise (see Table 2). These scores suggest that *type_features* are inherently well-structured, allowing for clear cluster boundaries. DBI and CHI metrics supported the quality of cluster separation, with low DBI values indicating minimal overlap between clusters.

Type-based clustering reflects inherent data structure and homogeneity in destination types, facilitating distinct grouping. The consistency across methods suggests that *type_features* can be clustered effectively with a range of techniques, although DBSCAN's efficiency and robustness against noise make it particularly advantageous for real-world applications.

4.1.4 Attr_Features

DBSCAN demonstrates excellent clustering performance on the *attr_features* dataset (Table 3), achieving a Sil_Score close to 1 with the Cityblock metric (0.99710), reflecting strong cohesion and separation between clusters. The extremely low DBI of 1.07E-08 also indicates clear boundaries between clusters. DBSCAN effectively handles noise, identifying only one point as noise, and results in very fast computation times, specifically 0.00050 seconds with the Cityblock metric, which provides the fastest computation among the tested metrics. Despite forming fewer clusters (5 clusters), DBSCAN produces compact and well-separated clusters.

In contrast, K-Means and FCM show similar performance but with some differences. K-Means, with 7 clusters, achieves a slightly higher Sil_Score (0.99712) compared to DBSCAN, but it has a longer computation time of 0.01568 seconds. FCM also results in a similar Sil_Score (0.99712) and the same number of clusters (7), but with a significantly faster computation time of 0.00001 seconds. However, both methods are less effective in handling noise, as all data points are assigned to clusters without distinguishing noise or outliers. As a result, even though both methods generate more clusters (7 clusters), their higher DBI values indicate less compact clusters compared to DBSCAN.

Thus, DBSCAN, with its density-based approach, proves superior in producing more distinct and compact clusters. While K-Means and FCM offer slight advantages in Sil_Score and computation time (particularly FCM with its very fast computation time), DBSCAN remains the better choice for handling data with noise and outliers, as it produces clusters with more defined boundaries.

4.1.5 Popularity_features

Clustering with *popularity_features*, which includes visit statistics and ratings, revealed clear separability across methods, with DBSCAN reaching a Sil_Score of 1 and forming three clusters with zero noise (Table 2). K-Means and FCM also achieved high-quality clustering, but DBSCAN's ability to handle sparse data effectively was evident in its superior performance.

The success of clustering *popularity_features* can be attributed to the high separability of the data, such as differences in visitor counts or rating distributions across destinations. DBSCAN's ability to handle data sparsity without compromising quality makes it particularly well-suited for clustering popularity based features.

4.1.6 Segment_features

For *segment_features*, which encompass visitor segmentation, DBSCAN performed exceptionally

well with a Sil_Score near 1, forming four clusters without noise (see Table 2). K-Means and FCM produced similar high scores, with results indicating clear cluster separation and minimal overlap.

Visitor segmentation provides distinct groupings, likely due to well-defined differences in segment characteristics (e.g., domestic vs. international visitor trends). DBSCAN's density-based approach, combined with the Cityblock metric, proved effective in managing segment data with minimal noise, making it suitable for applications requiring precise segmentation in tourism datasets.

4.1.7 Feature_combined

The *feature_combined* subset, which integrates multiple feature types, introduced significant clustering challenges. DBSCAN's Sil_Score dropped to 0.46700, forming 2 clusters with notable overlap, as reflected by higher DBI scores (Table 2). Both K-Means and FCM achieved similar scores, with substantial noise in the Hamming distance results (Sil_Score of -1), underscoring the complexity of clustering mixed data types.

Mixed feature types pose challenges due to variations in scale, distribution, and inherent data structure. The poor performance in the *feature_combined* subset suggests that traditional clustering methods are insufficient for complex multi-type data without significant pre-processing or dimensionality reduction.

4.2 Discussion

This section delves into the implications of distance metrics, the impact of *eps* and *minPts* parameters, computational efficiency, and a comprehensive comparison of clustering methods across different feature subsets.

4.2.1 Implications of distance metrics on DBSCAN

The choice of distance metric significantly impacts DBSCAN's clustering effectiveness, especially when feature characteristics vary across subsets. Different metrics have distinct effects on clustering quality, as clearly seen in the clustering results across different subsets (see Table 4).

The Cityblock showed excellent results for spatial data, as seen in the *geo_features* subset, as well as *popularity_features* subset. Sil_Scores were notably higher when the Cityblock metric was applied to these subsets, indicating its ability to capture proximity in spatial and density-based data. This metric is particularly effective when the distance between data points is defined by linear differences across dimensions, resulting in clearer and more cohesive clusters, especially for spatially structured or densely packed data.

On the other hand, the Cosine and Euclidean metrics performed exceptionally well on subsets like *cat_features* and *attr_features*. Both metrics achieved near-perfect Sil_Scores (close to 1), demonstrating their effectiveness for data where vector-based similarity is key. In addition, the low DBI values and high cluster cohesion highlight their ability to group data effectively, without significant overlap between clusters. These metrics are well-suited for data that is more vector-based, yielding clusters with clear boundaries between them.

Additionally, the Minkowski metric with p=3 also showed decent results, especially in more complex or mixed-data subsets like *feature_combined*. While it didn't perform as well as Cityblock or Cosine on certain subsets, Minkowski with p=3 provided more stable results compared to Hamming. It is somewhat sensitive to distances between data points that are farther apart, making the clustering results slightly more variable. This metric yielded moderate Sil_Scores and slightly higher DBI values compared to the more optimal metrics for specific data subsets.

Overall, the choice of distance metric significantly impacts DBSCAN's clustering quality. This influence becomes more pronounced in subsets with diverse data characteristics, emphasizing the importance of selecting the appropriate metric to achieve optimal clustering results.

4.2.2 Impact of eps and $minPts$ on DBSCAN

The performance of DBSCAN is highly influenced by the choice of its *eps* and *minPts* parameters, both of which play a crucial role in determining cluster density and noise management. These parameters significantly affect how DBSCAN identifies clusters, with different settings optimal for various feature subsets.

The *eps* parameter controls the size of the neighborhood around each point, and its effect is most noticeable in the density of the resulting clusters. Lower *eps* values were found to be optimal for denser clusters, as seen in subsets such as *popularity_features* and *segment_features*, where smaller *eps* values helped to minimize noise and produced higher Sil_Scores. On the other hand, higher *eps* values allowed DBSCAN to better handle sparse data, as evidenced in the geo_features subset, where larger *eps* values helped to identify clusters within the more dispersed data. However, higher *eps* values also carried the risk of merging distinct

clusters, which led to a reduction in the cohesion of the identified groups.

Similarly, the *minPts* parameter, which specifies the minimum number of points required to form a cluster, also plays a critical role in the clustering process. Higher *minPts* values increased DBSCAN's resilience to outliers, as they required more points to form clusters. This was particularly beneficial in the *cat_features* and *type_features* subsets, where higher *minPts* values, ranging between 20 to 23, improved the separation between clusters. Conversely, lower *minPts* values were more effective in creating smaller, denser clusters with minimal noise, particularly in subsets like *attr_features* and *popularity_features*, which typically contained more compact, well-separated data.

Table 5 summarizes the optimal *eps* and *minPts* values for each feature subset, along with the resulting Sil_Scores and additional notes on the effectiveness of the parameter settings for each type of data. The table demonstrates that DBSCAN's performance is heavily dependent on the specific characteristics of the data being analyzed, with different parameter configurations providing the best results depending on the density and nature of the feature subsets.

Table 4. Best distance metrics for DBSCAN performance

Feature subset	Best distance metric	Sil_Score	DBI	Notes
geo_features	Cityblock	0.54940	5.20E-01	Best metric for spatial separation
cat_features	Cosine, Euclidean	1	1.34E-08	Best for categorical data
type_features	Cityblock, Euclidean	0.99990	1.18E-08	Suitable for homogeneous data
attr_features	Euclidean, Cosine	0.99710	1.07E-08	Effective on attribute-based data
popularity_features	Cityblock	1	1.72E-08	Ideal for popularity-based data
segment_features	Cityblock, Cosine	1	8.11E-09	Clear cluster separation
feature_combined	Cosine	0.46700	3.92E-01	Mixed results due to data complexity

Table 5. Optimal DBSCAN parameters for each subset

Feature subset	Optimal parameter		Sil_Score	Notes
	eps	*minPts*		
geo_features	1.0	29	0.5494	Larger eps for spatial data
cat_features	3.1	20	1	High $minPts$ for categorical data
type_features	3.5	23	0.9999	Balanced eps and $minPts$
attr_features	2.2	2	0.9971	Small $minPts$ suits dense clusters
popularity_features	2.5	3	1	Smaller eps for dense data
segment_features	0.3	2	1	High density clusters
feature_combined	0.6	3	0.46700	Mixed characteristics require moderate eps

Table 6. Average computational time comparison across methods

Feature subset	Computational time (second)			Notes
	DBSCAN	K-Means	FCM	
geo_features	0.00740	0.01562	0.01130	DBSCAN fastest on spatial data
cat_features	0.00110	0.00001	0.00912	K-Means faster but less effective on noise
type_features	0.00030	0.01201	0.00348	DBSCAN effective on homogeneous data
attr_features	0.00050	0.01568	0.00001	High efficiency in dense, small clusters
popularity_features	0.00100	0.00110	0.00010	DBSCAN's density approach ideal
segment_features	0.00100	0.00929	0.01222	DBSCAN optimal for high-density clusters
feature_combined	0.01360	0.01156	0.00110	DBSCAN slower but gives the best clusters

Table 7. Comparative performance of DBSCAN, K-Means, and FCM on each feature subset

Feature subset	Comparative performance (Sil_Score; DBI)			Best method
	DBSCAN	K-Means	FCM	
geo_features	0.54940; 5.20E-01	0.48374; 6.98E-01	0.45724; 7.02E-01	DBSCAN
cat_features	1; 1.34E-08	1; 1.34E-08	1; 1.34E-08	All comparable
type_features	0.99990; 1.18E-08	1; 1.18E-08	1; 1.18E-08	All comparable
attr_features	0.99710; Near 0	0.99712; 1.07E-08	0.99712; 1.07E-08	All comparable
popularity_features	1; 1.72E-08	1; 1.72E-08	1; 1.72E-08	All comparable
segment_features	1; 8.11E-09	1; 8.11E-09	1; 8.11E-09	All comparable
feature_combined	0.46700; 3.92E-01	0.46440; 4.66E-01	0.04347; 1.91E+00	DBSCAN

4.2.3 Computational efficiency across methods

The computational efficiency of the clustering methods varied significantly across different feature subsets, especially when dealing with more complex data structures (Table 6). DBSCAN proved to be highly efficient for smaller and more homogeneous subsets, such as *attr_features* and *popularity_features*, where execution times were consistently below 0.01 seconds. This was due to DBSCAN's density-based approach, which is particularly effective for handling simpler, denser data.

However, for more complex data subsets, such as *feature_combined*, DBSCAN required longer processing times. The increased complexity of these subsets, which combined different types of data, led to more computationally intensive clustering tasks, as DBSCAN had to evaluate density-based regions and manage noise more effectively.

In comparison, K-Means displayed relatively consistent computational times across all subsets. Since K-Means relies on a centroid-based approach, its performance did not fluctuate as much depending on the subset. However, K-Means struggled more with noise and sparse clusters, which impacted the quality of the clusters it identified, even though the computational effort remained steady. The FCM method, was slower than DBSCAN across most feature subsets. This was especially true for categorical and mixed data types, where FCM's computational demands were higher due to the additional complexity introduced by membership calculations. These iterative processes required more computational resources and resulted in longer processing times compared to DBSCAN.

4.2.4 Comparison of clustering methods

A comparative analysis of DBSCAN, K-Means, and FCM reveals their respective strengths and weaknesses across different feature subsets. DBSCAN consistently outperformed both K-Means and FCM in handling subsets with varying densities, whether sparse or dense. It demonstrated a remarkable ability to manage noise and accurately identify clusters in data with varying densities. In particular, DBSCAN achieved higher Sil_Scores and lower DBI values for subsets like *geo_features* and *popularity_features*, reflecting its superior clustering performance and ability to handle complex data structures effectively.

On the other hand, K-Means performed adequately on subsets that exhibited more homogeneous structures, such as *type_features*. However, it struggled when dealing with noise and sparse data. This is because K-Means forces all data points into clusters, even when they may not fit well together. This often resulted in higher DBI values and greater overlap between clusters. The method's inability to adapt well to varying data densities led to less effective clustering in these cases.

FCM showed robust performance on subsets with continuous data distributions, but its membership approach was less effective for managing categorical or mixed data. In some cases, this approach led to overlapping clusters, as seen in the moderate Sil_Scores for *feature_combined* and *geo_features*. This suggests that while FCM works well with certain data types, its performance can be compromised when dealing with more complex or heterogeneous data structures.

The findings summarized in Table 7 provide an overview of the comparative performance of DBSCAN, K-Means, and FCM across each feature subset. The table highlights DBSCAN's strengths in managing noise, adapting to variations in density, and delivering higher-quality clustering results compared to the other two methods.

5. Conclusion

The optimized multi-clustering framework using DBSCAN, combined with various distance metrics, has proven highly effective in addressing the challenges of tourism data analysis. DBSCAN's ability to identify clusters with irregular shapes and manage noise consistently outperforms other methods, such as K-Means and FCM. This superiority is evident in its higher Sil_Scores and lower DBI values across subsets like *geo_features*, *popularity_features*, and *segment_features*.

The primary advantage of DBSCAN lies in its flexibility to handle density variations and complex cluster structures without requiring a predefined number of clusters. However, its performance is sensitive to parameters such as *eps* and *minPts*, necessitating careful tuning to achieve optimal outcomes. Furthermore, the selection of appropriate distance metrics significantly enhances clustering performance. For instance, the Cityblock metric is particularly effective for geographical data, while Euclidean and Cosine metrics perform better for attribute-based subsets.

In comparison, K-Means demonstrates efficiency in homogeneous datasets but struggles with noise and complex cluster shapes, often resulting in forced clusters and higher DBI scores. Similarly, FCM is more suitable for continuous data but encounters challenges when applied to mixed or categorical data,

leading to overlapping clusters and increased computational time.

DBSCAN also exhibits strong computational efficiency, particularly when applied to smaller and denser datasets, reinforcing its adaptability to large and complex tourism datasets. This study highlights the importance of selecting appropriate feature subsets, distance metrics, and parameter tuning to optimize clustering outcomes.

The findings of this study underscore the advantages of DBSCAN in analyzing tourism data, particularly in managing noise, density variations, and diverse cluster shapes. By leveraging DBSCAN's strengths and tailoring its parameters and metrics to the specific characteristics of tourism datasets, this research contributes significantly to the advancement of clustering methodologies and establishes DBSCAN as a robust tool for analyzing complex tourism data.

Conflicts of Interest

The authors declare there is no conflict of interest in this work.

Author Contributions

Conceptualization and methodology, SH, AM and EF; Implementation the of methodology, EF; data validation, AM and EF; formal analysis, SH, AM, and EF; investigation, EF, AM and SH; writing—original draft preparation, EF; writing—review and editing, SH, AM and EF; supervision, SH and AM; funding acquisition, EF.

Acknowledgments

The authors would like to express their gratitude to Universitas Teknologi Digital Indonesia for the funding support throughout this research and publication. We also thank the Department of Computer Science and Electronics, Universitas Gadjah Mada, for providing facilities and resources.

References

[1] S. J. Miah, H. Q. Vu, J. Gammack, and M. McGrath, "A Big Data Analytics Method for Tourist Behaviour Analysis", *Inf Manag*, Vol. 54, No. 6, pp. 771–785, 2017.

[2] W. Höpken, M. Fuchs, and M. Lexhagen, "Big Data Analytics for Tourism Destinations", In: *Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics*, pp. 28–45, 2019.

[3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", In: *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996.

[4] A. Fauzan, G. Fadillah, A. Fitria, H. Adriana, and M. Bariklana, "Cluster Mapping of Waste Exposure Using DBSCAN Approach: Study of Spatial Patterns and Potential Distribution in Bantul Regency", *Int J Informatics Vis*, Vol. 8, No. 2, pp. 751–759, 2024.

[5] M. Fuchs and W. Höpken, "Clustering", In: *Applied Data Science in Tourism*, 2022.

[6] J. W. Li, X. Q. Han, J. W. Jiang, Y. Hu, and L. Liu, "An Efficient Clustering Method for Dbscan Geographic Spatio-Temporal Large Data With Improved Parameter Optimization", *Int Arch Photogramm Remote Sens Spat Inf Sci - ISPRS Arch*, Vol. 42, No. 3/W10, pp. 581–584, 2020.

[7] P. D'Urso, L. De Giovanni, M. Disegna, R. Massari, and V. Vitale, "A Tourist Segmentation Based on Motivation, Satisfaction and Prior Knowledge with a Socio-Economic Profiling: A Clustering Approach with Mixed Information", *Soc Indic Res*, Vol. 154, No. 1, pp. 335–360, 2021.

[8] C. Santos-Mangudo and A. J. Heras, "A fair-multicluster approach to clustering of categorical data", *Cent Eur J Oper Res*, Vol. 31, No. 2, pp. 583–604, 2023.

[9] D. Scaldelai, L. C. Matioli, S. R. Santos, and M. Kleina, "MulticlusterKDE: a new algorithm for clustering based on multivariate kernel density estimation", *J Appl Stat*, Vol. 49, No. 1, pp. 98–121, 2022.

[10] C. Santos-Mangudo and A. J. Heras, "A Multicluster Approach to Selecting Initial Sets for Clustering of Categorical Data", *Interdiscip J Information, Knowledge, Manag*, Vol. 15, No. 1, pp. 227–246, 2020.

[11] D. Jain, M. Singh, and A. K. Sharma, "Performance enhancement of DBSCAN density based clustering algorithm in data mining", In: *Proc. of 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 1559–1564, 2017.

[12] Y. Xie, X. Jia, S. Shekhar, H. Bao, and X. Zhou, "Significant DBSCAN+: Statistically Robust Density-based Clustering", *ACM Trans Intell Syst Technol*, Vol. 12, No. 5, pp. 1–26, 2021.

[13] G. Al-Naymat, M. Khader, M. A. Al-Betar, R. Hriez, and A. Hadi, "MR-VDENCLUE: Varying Density Clustering Using MapReduce", *Intelligent Systems with Applications*, pp. 771–

788, 2023.
[14] F. Ozge Ozkok, "A New Approach to Determine Eps Parameter of DBSCAN Algorithm", *Int J Intell Syst Appl Eng*, Vol. 5, No. 4, p. 247, 2017.
[15] X. Zhang and S. Zhou, "WOA-DBSCAN: Application of Whale Optimization Algorithm in DBSCAN Parameter Adaption", *IEEE Access*, Vol. 11, pp. 91861–91878, 2023.
[16] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis", *J Comput Appl Math*, Vol. 20, No. C, pp. 53–65, 1987.
[17] T. Caliński and J. Harabasz, "A Dendrite Method Foe Cluster Analysis", *Commun Stat*, Vol. 3, No. 1, pp. 1–27, 1974.
[18] C. Rashmi and M. M. Kodabagi, "QST-ClustNet: Quantum Inspired Sooty Tern Optimization for Clustering User Profiles in Social Network", *Int J Intell Eng Syst*, Vol. 16, No. 4, pp. 537–547, 2023.
[19] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure", *IEEE Trans Pattern Anal Mach Intell*, Vol. PAMI-1, No. 2, pp. 224–227, 1979.
[20] B. Rawat and S. K. Dwivedi, "Analyzing the Performance of Various Clustering Algorithms", *Int J Mod Educ Comput Sci*, Vol. 11, No. 1, pp. 45–53, 2019.
[21] R. Hidayat, A. Harjoko, and A. Musdholifah, "A Robust Image Retrieval Method Using Multi-Hierarchical Agglomerative Clustering and Davis-Bouldin Index", *Int J Intell Eng Syst*, Vol. 15, No. 2, pp. 441–453, 2022.
[22] R. S. M. Lakshmi Patibandla and V. N, "Clustering Algorithms: An Exploratory Review", *Data Clustering*, Vol. 10, 2022.
[23] H. Yin, A. Aryani, S. Petrie, A. Nambissan, A. Astudillo, and S. Cao, "A Rapid Review of Clustering Algorithms", *arXiv:2401.07389*, pp. 1–25, 2024.
[24] J. Wang, C. Zhu, Y. Zhou, X. Zhu, Y. Wang, and W. Zhang, "From Partition-Based Clustering to Density-Based Clustering: Fast Find Clusters With Diverse Shapes and Densities in Spatial Databases", *IEEE Access*, Vol. 6, pp. 1718–1729, 2018.
[25] P. Ghosh and S. Mukherjee, "Understanding tourist behaviour towards destination selection based on social media information: an evaluation using unsupervised clustering algorithms", *J Hosp Tour Insights*, Vol. 6, No. 2, pp. 754–778, 2023.
[26] J. A. Orama, A. Huertas, J. Borràs, A. Moreno, and S. Anton Clavé, "Identification of Mobility Patterns of Clusters of City Visitors: An Application of Artificial Intelligence Techniques to Social Media Data", *Appl Sci*, Vol. 12, No. 12, p. 5834, 2022.
[27] A. Starczewski, P. Goetzen, and M. J. Er, "A New Method for Automatic Determining of the DBSCAN Parameters", *J Artif Intell Soft Comput Res*, Vol. 10, No. 3, pp. 209–221, 2020.
[28] G. G. Ladha and R. K. S. Pippal, "An efficient distance estimation and centroid selection based on k-means clustering for small and large dataset", *Int J Adv Technol Eng Explor*, Vol. 7, No. 73, pp. 234–240, 2020.
[29] A. Fahim, "An Extended DBSCAN Clustering Algorithm", *Int J Adv Comput Sci Appl*, Vol. 13, No. 3, pp. 245–258, 2022.
[30] C. Li, L. Gao, Y. Liu, and H. Li, "Cluster analysis of China's inbound tourism market: A new multi-attribute approach based on association rule mining of tourist preferences at scenic spots", *Asia Pacific J Tour Res*, Vol. 26, No. 6, pp. 654–667, 2021.
[31] M. Mittal, L. M. Goyal, D. J. Hemanth, and J. K. Sethi, "Clustering approaches for high-dimensional databases: A review", *WIREs Data Min Knowl Discov*, Vol. 9, No. 3, pp. 647–666, 2019.