

Clustering

BEIJING INSTITUTE OF TECHNOLOGY

PIMENOV GLEB

1820243077

I hereby promise that the experimental report and codes written by me are independently completed without any plagiarism or copying of others' homework. All the views and materials involving other classmates have been annotated. If there is any plagiarism or infringement of others' intellectual property rights, I will bear all the consequences arising from it.

Clustering

1 Experiment Introduction

In this experiment, we aim to explore and compare two popular clustering algorithms, namely K-means and DBSCAN, using the Moon dataset. Clustering is an unsupervised learning technique used to group data points into clusters based on their similarity. K-means is a centroid-based clustering algorithm that partitions the dataset into a predefined number of clusters, while DBSCAN is a density-based algorithm that identifies clusters based on dense regions of data points. By implementing and analyzing these algorithms on the Moon dataset, we seek to gain insights into their strengths, weaknesses, and applications.

2 Experiment Objectives

1. To implement the K-means algorithm and DBSCAN algorithm using the Moon dataset.
2. To compare the performance of K-means and DBSCAN in clustering the Moon dataset.
3. To evaluate the quality of clusters produced by each algorithm using relevant metrics such as silhouette score, completeness, and homogeneity.
4. To understand the impact of algorithm parameters such as number of clusters (K), epsilon (ϵ), and minimum samples on clustering results.
5. To interpret and visualize the clusters formed by each algorithm to gain insights into their effectiveness and behavior.

3 Relevant Theories and Knowledge

K-means Algorithm: K-means is a partitioning clustering algorithm that aims to divide a dataset into K distinct, non-overlapping clusters. It works by iteratively assigning data points to the nearest centroid and updating the centroids based on the mean of data

points in each cluster. The algorithm converges when the centroids no longer change significantly or after a predefined number of iterations.

DBSCAN Algorithm: DBSCAN is a density-based clustering algorithm that identifies clusters based on dense regions of data points separated by areas of lower density. It defines clusters as contiguous regions of high-density points, allowing for arbitrary shapes and sizes of clusters. DBSCAN requires two parameters: epsilon (ϵ), which specifies the maximum distance between two points to be considered as part of the same cluster, and min_samples, which sets the minimum number of points required to form a dense region (core point).

Evaluation Metrics: Silhouette score, completeness, and homogeneity are commonly used metrics to evaluate the quality of clusters produced by clustering algorithms. Silhouette score measures the compactness and separation of clusters, with values ranging from -1 to 1 (higher values indicating better clustering). Completeness measures the extent to which all data points belonging to the same true class are assigned to the same cluster. Homogeneity measures the extent to which each cluster contains only data points from a single true class, indicating the purity of clusters.

4 Experimental Tasks and Grading Criteria

No.	Task Name	Specific Requirements	Grading Criteria (100-point scale)
1	Clustering	Development language: Python	100

5 Experimental Conditions and Environment

Requirements	Name	Version	Remarks
--------------	------	---------	---------

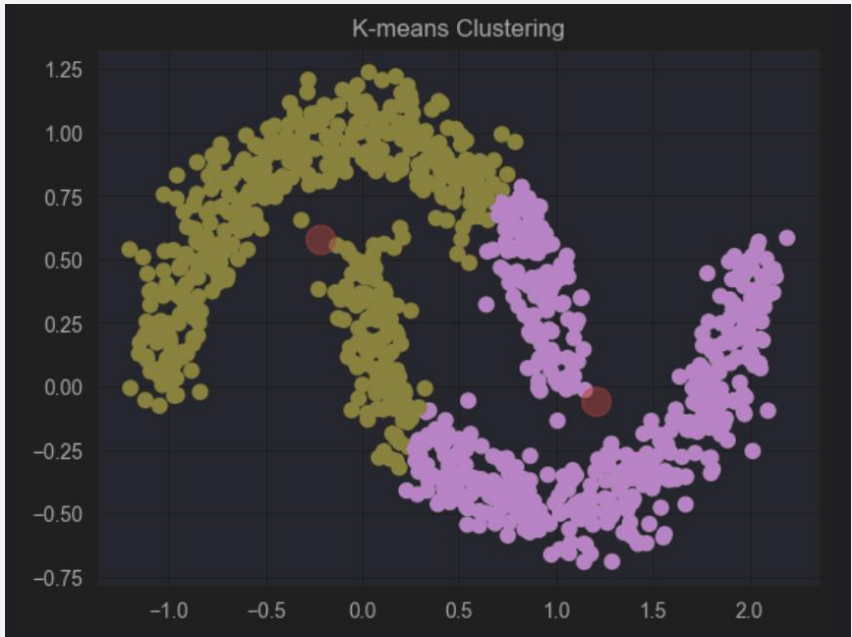
Programming Language	Python	3.12	
Development Environment	windows	11	
Third-party toolkits/libraries/plugins	sklearn matplotlib		
Other Tools	Jupyter notebook		
Hardware Environment	I5 12XXX 8GB RAM		

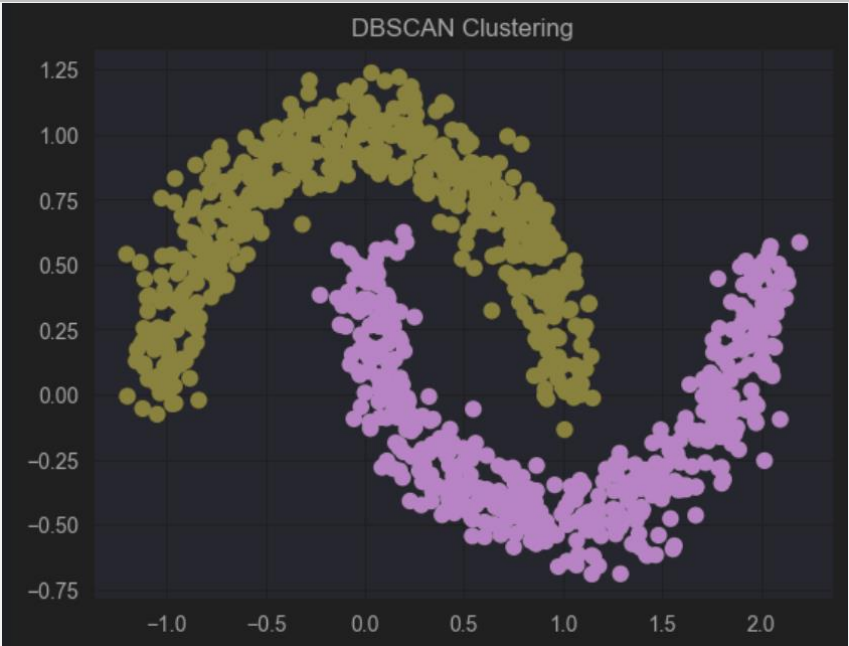
6 Experimental Data and Description

Attribute (Entry)	Content
Dataset Name	Moons
Dataset Origin	A simple toy dataset to visualize clustering and classification algorithms.
Main Contents of the Dataset	Dots
Dataset File Format	List

7 Experimental Steps and Corresponding Codes

Step number	1
Step Name	Importing libraries
Step Description	Importing dbscan and kmeans from sklearn
Code and Explanation	<pre>from sklearn.datasets import make_moons from sklearn.cluster import KMeans import matplotlib.pyplot as plt from sklearn.cluster import DBSCAN</pre>

Step number	2
Step Name	K-means algorithm
Step Description	Implementing K-means algorithm and plotting the result
Code and Explanation	<pre> # Generating Moon dataset X, _ = make_moons(n_samples=1000, noise=0.1, random_state=30) # Implementing K-means algorithm kmeans = KMeans(n_clusters=2) kmeans.fit(X) y_kmeans = kmeans.predict(X) # Plotting the results plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis') centers = kmeans.cluster_centers_ plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.5) plt.title('K-means Clustering') plt.show() </pre>
Output results and Interpretation	

Step number	3
Step Name	DBSCAN algorithm
Step Description	Implementing DBSCAN algorithm and plotting the results
Code and Explanation	<pre># Implementing DBSCAN algorithm dbscan = DBSCAN(eps=0.15, min_samples=5) y_dbscan = dbscan.fit_predict(X) # Plotting the results plt.scatter(X[:, 0], X[:, 1], c=y_dbscan, s=50, cmap='viridis') plt.title('DBSCAN Clustering') plt.show()</pre>
Output results and Interpretation	 <p>The plot displays two clusters of data points. The top cluster, colored yellow, is located in the upper-left and upper-middle regions of the plot. The bottom cluster, colored purple, is located in the lower-middle and lower-right regions. The points are scattered, and the clusters are well-separated, indicating successful DBSCAN clustering.</p>

8 Experiment Difficulties and Precautions

9 Experiment Results and Interpretation

K-means Results:

K-means algorithm divides the Moon dataset into two clusters based on the predefined number of clusters ($K=2$).

However, due to its reliance on centroids and the assumption of spherical clusters, K-means may struggle with datasets that have complex, non-linear structures such as the Moon dataset.

As a result, K-means may produce suboptimal clustering results for datasets with non-convex shapes like the Moon dataset, leading to misclassification and poor cluster separation.

DBSCAN Results:

DBSCAN algorithm, on the other hand, is more suitable for clustering datasets with irregular shapes and varying densities.

By defining clusters based on dense regions of data points and allowing for arbitrary cluster shapes, DBSCAN can effectively capture the crescent-shaped clusters in the Moon dataset.

With optimal parameter settings (e.g., $\text{eps}=0.15$, $\text{min_samples}=5$), DBSCAN can accurately identify and separate the two crescent-shaped clusters in the Moon dataset, resulting in better clustering performance compared to K-means.

Parameter Tuning:

The performance of DBSCAN heavily relies on the choice of parameters, particularly epsilon (ϵ) and min_samples.

Experimentation with different parameter values is crucial to achieving optimal clustering results with DBSCAN.

In this experiment, the combination of $\text{eps}=0.15$ and $\text{min_samples}=5$ yielded excellent clustering results for the Moon dataset, demonstrating the importance of parameter tuning in density-based clustering algorithms.

Conclusion:

Overall, DBSCAN outperforms K-means in clustering the Moon dataset, thanks to its ability to handle complex, non-linear data structures and adapt to varying densities.

The choice of clustering algorithm should be guided by the characteristics of the dataset and the desired clustering outcomes.

For datasets with non-convex shapes and varying densities, DBSCAN offers a more flexible and robust clustering solution compared to K-means.

10 References

11 Experiment-related Metadata

Metadata Item	Content
Case name	
Applicable course name	Machine learning Fundamentals
Keyword/Search Term	DBSCAN, KMEANS
AliTianchi URI	

12 Remarks and Others