

DWM\_Experiment no\_04

AIM: Implementation of Clustering Algorithm (K-means / Agglomerative) usingPython

**Introduction:** K-means clustering is a centroid-based method where the data points are grouped around a predefined number of clusters (K). The algorithm aims to minimize the variance within each cluster by iteratively assigning points to the nearest cluster center and adjusting the centers accordingly.

\*Procedure \*

- 1. Import Required Libraries: Use numpy, matplotlib, and sklearn.
- 2. Generate Sample Data: Create synthetic data using make\_blobs().
- 3. Visualize Data: Plot the generated dataset before clustering.
- 4. Apply K-means Clustering: o Initialize K-means with a chosen number of clusters (n\_clusters).  
o Fit the model to the dataset. o Retrieve cluster centers and assigned labels.
- 5. Visualize Clusters: Plot clusters with different colors and highlight centroids.
- 6. Evaluate Performance: Compare results and analyze the cluster separation.

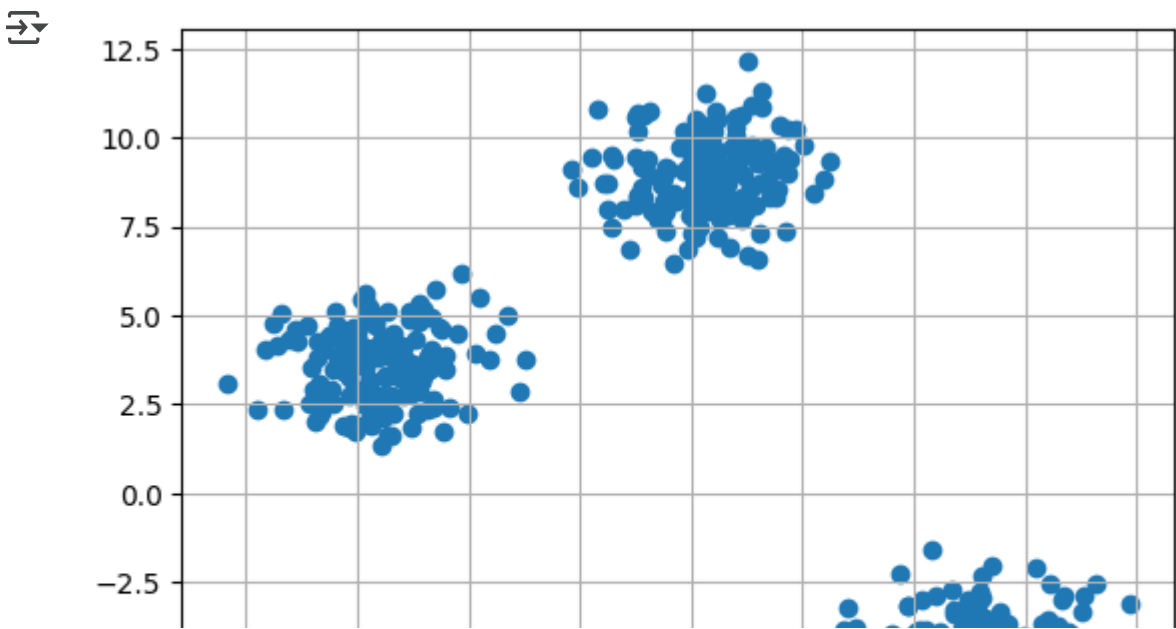
\*K-means Clustering \*

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

X,y = make_blobs(n_samples = 500,n_features = 2,centers = 3,random_state = 23)

fig = plt.figure(0)
plt.grid(True)
plt.scatter(X[:,0],X[:,1])
plt.show()
```



```
k = 3
```

```
clusters = {}  
np.random.seed(23)
```

```
for idx in range(k):  
    center = 2*(2*np.random.random((X.shape[1],))-1)  
    points = []  
    cluster = {  
        'center' : center,  
        'points' : []  
    }  
  
    clusters[idx] = cluster
```

```
clusters
```

```
⇒ {0: {'center': array([0.06919154, 1.78785042]), 'points': []},  
   1: {'center': array([ 1.06183904, -0.87041662]), 'points': []},  
   2: {'center': array([-1.11581855,  0.74488834]), 'points': []}}
```

```
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
from sklearn.datasets import make_blobs
```

```
# Generate sample data for clustering  
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)
```


```
# Visualize the data  
plt.scatter(X[:, 0], X[:, 1], s=50, cmap='viridis')  
plt.title("Generated Data for Clustering")  
plt.show()
```

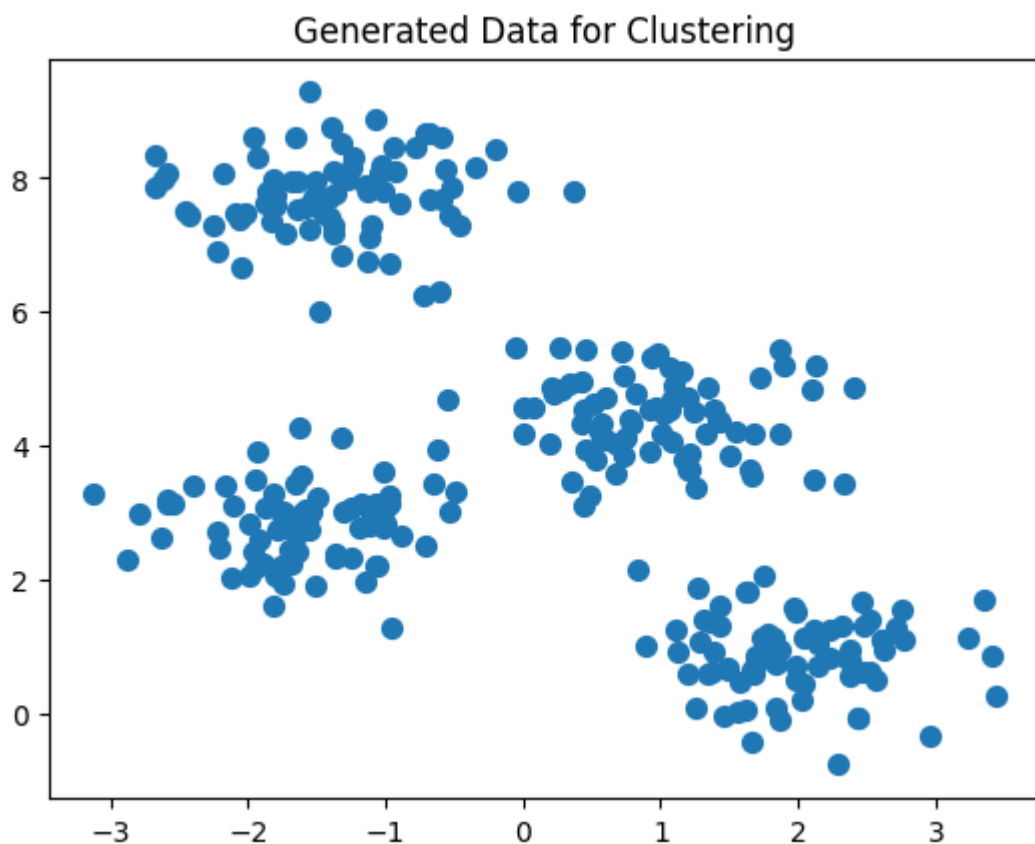
```
# Apply KMeans clustering  
kmeans = KMeans(n_clusters=4) # Let's choose 4 clusters  
kmeans.fit(X)
```

```
# Get the cluster centers (centroids)  
centers = kmeans.cluster_centers_
```

```
# Get the cluster labels (which cluster each point belongs to)  
labels = kmeans.labels_
```

```
# Visualize the clusters  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X') # Mark the  
plt.title("K-means Clustering Results")  
plt.show()
```

 <ipython-input-6-653dc924e10b>:10: UserWarning: No data for colormapping provided  
plt.scatter(X[:, 0], X[:, 1], s=50, cmap='viridis')



## Review Questions

### 1. What is the K-means clustering algorithm, and how does it work?

K-means is an iterative clustering algorithm that partitions data into K clusters by following these steps:

1. Initialize K cluster centroids randomly.
2. Assign each data point to the nearest centroid.
3. Compute new centroids as the mean of all assigned points.
4. Repeat steps 2-3 until centroids stabilize or reach the maximum iterations.

## 2. How do you determine the optimal number of clusters in K-means?

- Elbow Method: Plot the Within-Cluster Sum of Squares (WCSS) against the number of clusters. The "elbow" point suggests the best K.
- Silhouette Score: Measures how well points fit within their assigned clusters. Higher scores indicate better clustering.
- Gap Statistic: Compares K-means clustering with a random clustering reference.

## 3. What are the common distance metrics used in Agglomerative Clustering?

Agglomerative Clustering uses different distance metrics to determine the proximity of clusters:

- Euclidean Distance (Most common)
- Manhattan Distance
- Cosine Similarity
- Mahalanobis Distance

## Conclusion:

The implementation of K-means clustering successfully grouped similar data points based on their features. By using centroid-based partitioning, the algorithm efficiently categorized the dataset into four clusters. The visualization confirmed well-defined clusters with distinct centroids.

Choosing the optimal number of clusters is crucial for accurate results, which can be determined using the Elbow Method or Silhouette Score.

Clustering is widely used in image segmentation, customer segmentation, anomaly detection, and pattern recognition. Understanding both K-means and Agglomerative Clustering helps in selecting the right method based on the dataset characteristics and application needs.

GITHUB : <https://github.com/panchaldeep1123/dwm>