
✅ **Problem Statement:**

We are given a dataset with unlabeled data points. The goal is to group these data points into clusters based on similarity using the **K-Means clustering algorithm**, and visualize both the original and clustered data.

💡 **Step-by-Step Approach:**

---

1. Understand the Algorithm (K-Means)

K-Means is an unsupervised clustering algorithm that:

- Divides the data into *K* clusters.
- Minimizes the **intra-cluster variance**.
- Works by:
    1. Randomly initializing *K* centroids.
    2. Assigning each point to the nearest centroid.
    3. Updating centroids based on the mean of assigned points.
    4. Repeating steps 2-3 until convergence.

---

2. Data Generation

```
from sklearn.datasets import make_blobs


X, y_true = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)
```

- make_blobs() generates synthetic data for clustering.
- n_samples=300: generate 300 points.
- centers=4: 4 clusters (ground truth).
- cluster_std=0.60: spread of each cluster.
- random_state=0: for reproducibility.

---

3. Visualize the Unlabeled Data

```
import matplotlib.pyplot as plt


plt.scatter(X[:, 0], X[:, 1], s=30)
```

plt.title("Randomly Generated Data")

plt.show()

- X[:, 0] and X[:, 1]: 2D features for each point.
- Gives a visual idea of how points might be clustered.

---

4. Apply K-Means Clustering

from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=4, random_state=0)

kmeans.fit(X)

y_kmeans = kmeans.predict(X)

- n_clusters=4: expected number of clusters.
- fit(X): computes clusters.
- predict(X): assigns each point to a cluster.
- y_kmeans: the predicted cluster labels.

---

5. Visualize the Clustering Result

plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=30, cmap='viridis')

centers = kmeans.cluster_centers_

plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.6, marker='X')

plt.title("K-Means Clustering")

plt.show()

- Color (c=y_kmeans) shows the cluster each point belongs to.
- kmeans.cluster_centers_: coordinates of centroids.
- Plotting these with a black "X" highlights the cluster centers.

📈 **Evaluation (Optional)**

Even though K-Means is unsupervised, you can evaluate clustering using:

from sklearn.metrics import adjusted_rand_score

```
print(adjusted_rand_score(y_true, y_kmeans))
```

- Since you have y_true from make_blobs(), you can compare.
- adjusted_rand_score ranges from -1 to 1 (1 = perfect match).

🔁 Optional Extensions

- **Elbow Method** to find optimal number of clusters.
- **Silhouette Score** for cluster quality.
- Apply to **real datasets** (e.g., Iris, Mall Customer data).
- Try **PCA** for high-dimensional data visualization.

🧠 Summary

- **K-Means** clusters data by minimizing variance.
- make_blobs is handy for testing clustering algorithms.
- Visualizations help understand clustering effectiveness.
- You can evaluate clustering performance when true labels are known (like in this example).
