12. Implement the K-means algorithm and apply it to the data you selected. Evaluate performance by measuring the sum of the Euclidean distance of each example from its class center. Test the performance of the algorithm as a function of the parameters K.

**Description:** K-Means is an unsupervised machine learning algorithm used for clustering data into K groups. It is commonly used in data analysis and pattern recognition to identify similar data points and group them together.

**The K-Means Works as follows:**

1. **Choose the Number of Clusters (K):**  
   The user selects the number of clusters, **K**, which represents the number of groups into which the data should be classified.
2. **Initialize Cluster Centers:**
   * Randomly select **K** points from the dataset as the initial **centroids (cluster centers)**.
   * Alternatively, the **K-Means++** initialization method selects smarter initial centroids to improve convergence.
3. **Assign Points to Nearest Cluster:**
   * For each data point, compute the **Euclidean distance** to each centroid.
   * Assign the point to the cluster with the **closest centroid**.
4. **Recompute Cluster Centroids:**
   * Calculate the **mean** of all data points in each cluster.
   * Update the **centroids** with these newly computed means.
5. **Repeat Until Convergence:**
   * Steps 3 and 4 are repeated **iteratively** until centroids **stop changing** significantly or the **maximum number of iterations** is reached.

The **Elbow Method** is commonly used to determine the best value of **K** by plotting the **sum of squared distances (inertia)** against different values of K. The "elbow" in the curve indicates the optimal number of clusters.

**Python code**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.cluster import KMeans**

**from sklearn.datasets import make\_blobs**

**from sklearn.metrics import pairwise\_distances\_argmin\_min**

**# Generate synthetic dataset**

**np.random.seed(42)**

**X, \_ = make\_blobs(n\_samples=300, centers=4, cluster\_std=1.0, random\_state=42)**

**# Function to compute sum of Euclidean distances from cluster centers**

**def evaluate\_kmeans(X, k\_values):**

**distortions = []**

**for k in k\_values:**

**kmeans = KMeans(n\_clusters=k, random\_state=42, n\_init=10)**

**kmeans.fit(X)**

**cluster\_centers = kmeans.cluster\_centers\_**

**labels = kmeans.labels\_**

**# Compute sum of Euclidean distances**

**\_, distances = pairwise\_distances\_argmin\_min(X, cluster\_centers)**

**total\_distance = np.sum(distances)**

**distortions.append(total\_distance)**

**print(f'K={k}, Total Distance: {total\_distance:.4f}')**

**return distortions**

**# Test K values from 1 to 10**

**k\_values = range(1, 11)**

**distortions = evaluate\_kmeans(X, k\_values)**

**# Plot the Elbow Method**

**plt.figure(figsize=(8, 5))**

**plt.plot(k\_values, distortions, marker='o', linestyle='--')**

**plt.xlabel('Number of Clusters (K)')**

**plt.ylabel('Sum of Euclidean Distances')**

**plt.title('Elbow Method for Optimal K')**

**plt.show()**

**Output:**