### K-Means Algorithm

**Program:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.cluster import KMeans**

**from sklearn.mixture import GaussianMixture**

**# Generate synthetic data**

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

**X1 = np.random.normal(loc=[2, 2], scale=0.5, size=(100, 2))**

**X2 = np.random.normal(loc=[8, 8], scale=0.5, size=(100, 2))**

**X3 = np.random.normal(loc=[2, 8], scale=0.5, size=(100, 2))**

**X = np.vstack((X1, X2, X3)) # Combine into one dataset**

**# Step 1: K-Means Initialization**

**k = 3 # Number of clusters**

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

**kmeans.fit(X)**

**kmeans\_labels = kmeans.labels\_**

**# Step 2: Expectation-Maximization (Gaussian Mixture Model)**

**gmm = GaussianMixture(n\_components=k, covariance\_type='full', random\_state=42)**

**gmm.fit(X)**

**gmm\_labels = gmm.predict(X)**

**# Plot Results**

**fig, axes = plt.subplots(1, 2, figsize=(12, 5))**

**# Plot K-Means Clustering**

**axes[0].scatter(X[:, 0], X[:, 1], c=kmeans\_labels, cmap='viridis', marker='o')**

**axes[0].scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], c='red', marker='x', s=100, label='Centroids')**

**axes[0].set\_title("K-Means Clustering")**

**axes[0].legend()**

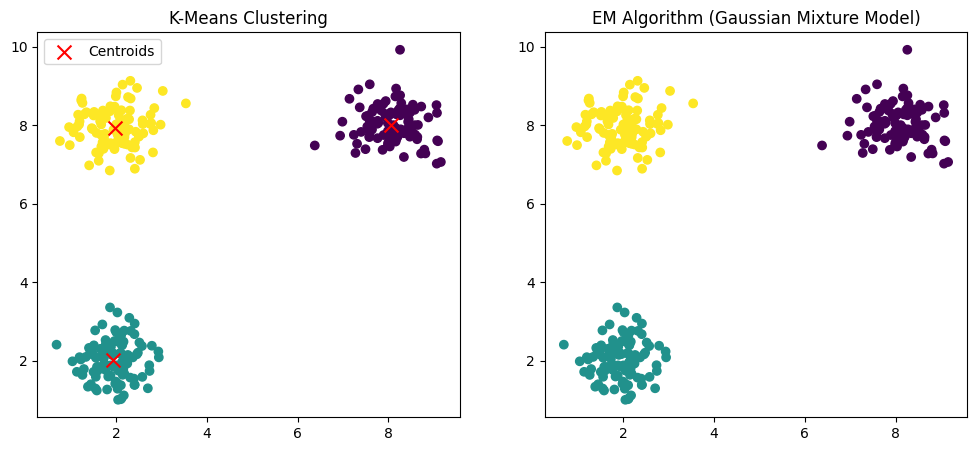
**# Plot GMM Clustering**

**axes[1].scatter(X[:, 0], X[:, 1], c=gmm\_labels, cmap='viridis', marker='o')**

**axes[1].set\_title("EM Algorithm (Gaussian Mixture Model)")**

**plt.show()**

**Output:**



# KNN On the Iris Dataset

**Program:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn import datasets**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.preprocessing import StandardScaler**

**from sklearn.neighbors import KNeighborsClassifier**

**from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix**

**# Load the Iris dataset**

**iris = datasets.load\_iris()**

**X = iris.data # Features: Sepal length, Sepal width, Petal length, Petal width**

**y = iris.target # Class labels: 0 (Setosa), 1 (Versicolor), 2 (Virginica)**

**# Split dataset into training and testing sets (80% training, 20% testing)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**# Normalize the data (optional but improves performance)**

**scaler = StandardScaler()**

**X\_train = scaler.fit\_transform(X\_train)**

**X\_test = scaler.transform(X\_test)**

**# Choose the number of neighbors (K)**

**k = 5**

**# Train the KNN classifier**

**knn = KNeighborsClassifier(n\_neighbors=k)**

**knn.fit(X\_train, y\_train)**

**# Make predictions**

**y\_pred = knn.predict(X\_test)**

**# Evaluate the model**

**accuracy = accuracy\_score(y\_test, y\_pred)**

**print(f"Accuracy: {accuracy \* 100:.2f}%")**

**print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))**

**# Confusion Matrix**

**conf\_matrix = confusion\_matrix(y\_test, y\_pred)**

**print("\nConfusion Matrix:\n", conf\_matrix)**

**# Visualize predictions (only for 2D projections)**

**plt.figure(figsize=(8, 6))**

**plt.scatter(X\_test[:, 0], X\_test[:, 1], c=y\_pred, cmap='viridis', edgecolors='k', marker='o', s=100)**

**plt.title(f'KNN Classification (K={k}) on Iris Dataset')**

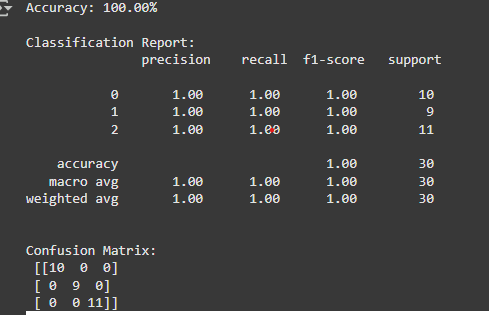
**plt.xlabel('Feature 1 (Sepal Length)')**

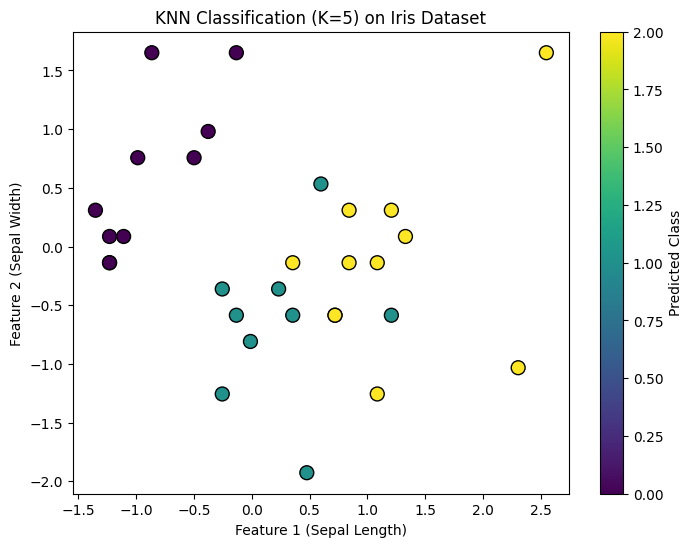
**plt.ylabel('Feature 2 (Sepal Width)')**

**plt.colorbar(label="Predicted Class")**

**plt.show()**

**OUTPUT:**





**Required Dependencies:**

**pip install numpy matplotlib scikit-learn**