**8.Implementation of K-mean algorithm**

**Aim:** To Implement K-mean algorithm

**Code:**

import numpy as np

import matplotlib.pyplot as plt

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

def euclidean\_distance(point1, point2):

return np.sqrt(np.sum((point1 - point2)\*\*2))

def kmeans(X, n\_clusters, max\_iters=100):

random\_indices = np.random.choice(X.shape[0], n\_clusters, replace=False)

centroids = X[random\_indices]

for \_ in range(max\_iters):

labels = []

for point in X:

distances = [euclidean\_distance(point, centroid) for centroid in centroids]

closest\_centroid\_index = np.argmin(distances)

labels.append(closest\_centroid\_index)

labels = np.array(labels)

new\_centroids = np.zeros\_like(centroids)

for i in range(n\_clusters):

points\_in\_cluster = X[labels == i]

if len(points\_in\_cluster) > 0:

new\_centroids[i] = np.mean(points\_in\_cluster, axis=0)

else:

new\_centroids[i] = X[np.random.choice(X.shape[0])]

if np.allclose(centroids, new\_centroids):

break

centroids = new\_centroids

return labels, centroids

X, \_ = make\_blobs(n\_samples=300, centers=3, cluster\_std=0.5, random\_state=42)

n\_clusters = 3

labels, centroids = kmeans(X, n\_clusters)

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

plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=50, alpha=0.8)

plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='X', s=200, label='Centroids')

plt.title('K-Means Clustering')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.grid(True)

plt.show()

def kmeans(X, n\_clusters, max\_iters=100):

random\_indices = np.random.choice(X.shape[0], n\_clusters, replace=False)

centroids = X[random\_indices]

for \_ in range(max\_iters):

labels = []

for point in X:

distances = [euclidean\_distance(point, centroid) for centroid in centroids]

closest\_centroid\_index = np.argmin(distances)

labels.append(closest\_centroid\_index)

labels = np.array(labels)

new\_centroids = np.zeros\_like(centroids)

for i in range(n\_clusters):

points\_in\_cluster = X[labels == i]

if len(points\_in\_cluster) > 0:

new\_centroids[i] = np.mean(points\_in\_cluster, axis=0)

else:

new\_centroids[i] = X[np.random.choice(X.shape[0])]

if np.allclose(centroids, new\_centroids):

break

centroids = new\_centroids

return labels, centroids

X, \_ = make\_blobs(n\_samples=300, centers=3, cluster\_std=0.5, random\_state=42)

n\_clusters = 3

labels, centroids = kmeans(X, n\_clusters)

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

plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=50, alpha=0.8)

plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='X', s=200, label='Centroids')

plt.title('K-Means Clustering')

plt.xlabel('Feature 1')

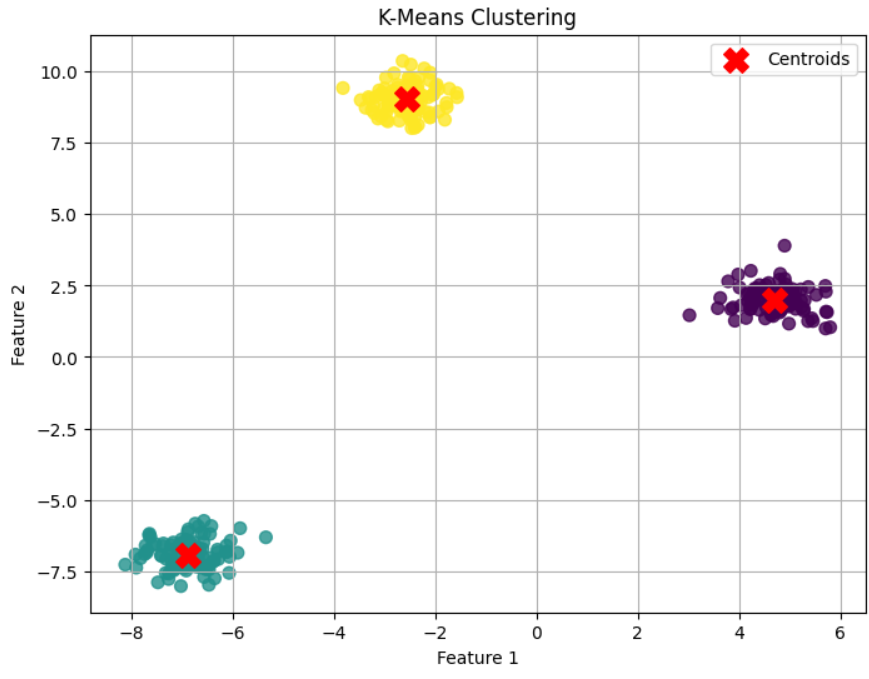
plt.ylabel('Feature 2')

plt.legend()

plt.grid(True)

plt.show()

**OUT PUT:**

****