**Experiment-11**

AIM: Apply EM algorithm to cluster a set of data stored in a. CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of the set two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.mixture import GaussianMixture

from matplotlib.patches import Ellipse

X, y\_true = make\_blobs(n\_samples=100, centers=4, cluster\_std=0.60, random\_state=0)

X = X[:, ::-1]

gmm = GaussianMixture(n\_components=4).fit(X)

labels = gmm.predict(X)

plt.scatter(X[:, 0], X[:, 1], c=labels, s=40, cmap='viridis')

probs = gmm.predict\_proba(X)

print(probs[:5].round(3))

size = 50 \* probs.max(1) \*\* 2

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

def draw\_ellipse(position, covariance, ax=None, \*\*kwargs):

ax = ax or plt.gca()

if covariance.shape == (2, 2):

U, s, Vt = np.linalg.svd(covariance)

angle = np.degrees(np.arctan2(U[1, 0], U[0, 0]))

width, height = 2 \* np.sqrt(s)

else:

angle = 0

width, height = 2 \* np.sqrt(covariance)

for nsig in range(1, 4):

ax.add\_patch(Ellipse(position, nsig \* width, nsig \* height, angle, \*\*kwargs))

def plot\_gmm(gmm, X, label=True, ax=None):

ax = ax or plt.gca()

if label:

labels = gmm.fit(X).predict(X)

ax.scatter(X[:, 0], X[:, 1], c=labels, s=40, cmap='viridis', zorder=2)

else:

ax.scatter(X[:, 0], X[:, 1], s=40, zorder=2)

ax.axis('equal')

w\_factor = 0.2 / gmm.weights\_.max()

for pos, covar, w in zip(gmm.means\_, gmm.covariances\_, gmm.weights\_):

draw\_ellipse(pos, covar, alpha=w \* w\_factor)

gmm = GaussianMixture(n\_components=4, random\_state=42)

plot\_gmm(gmm, X)

gmm = GaussianMixture(n\_components=4, covariance\_type='full', random\_state=42)

plot\_gmm(gmm, X)

plt.show()

Output:

[[0. 0. 1. 0.]

[0. 1. 0. 0.]

[0. 0. 1. 0.]

[0. 0. 1. 0.]

[0. 0. 1. 0.]]

