**Ex.No. 5**

***Write a program to implement the Gaussian mixture model. Compute the accuracy of the classifier, considering few test data sets.***

***Tools: Rapid Miner, Python, Scikitlearn, Anaconda navigator***

**Algorithm**

1. Initialize the mean μk, the covariance matrix μk, and the mixing coefficients μk by some random variables.
2. Compute the μk
3. Again, estimate all the parameters using the current μk values
4. Compute log-likelihood function.
5. Put some convergence criterion

If the log-likelihood value converges to some value (or if all the parameters converge to some values) then stop, else return to Step 2.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.mixture import GaussianMixture

from sklearn.datasets import load\_iris

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

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.metrics import accuracy\_score, confusion\_matrix

# -----------------------------

# 1. Load dataset

# -----------------------------

iris = load\_iris()

X = iris.data

y = iris.target

target\_names = iris.target\_names

# -----------------------------

# 2. Standardize data

# -----------------------------

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# -----------------------------

# 3. Split into training and test sets

# -----------------------------

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X\_scaled, y, test\_size=0.2, random\_state=42, stratify=y

)

# -----------------------------

# 4. Train GMM model

# -----------------------------

gmm = GaussianMixture(n\_components=3, random\_state=42)

gmm.fit(X\_train)

# Predict clusters

y\_train\_pred = gmm.predict(X\_train)

y\_test\_pred = gmm.predict(X\_test)

# -----------------------------

# 5. Map GMM clusters to real classes

# -----------------------------

cluster\_map = {}

for cluster in np.unique(y\_train\_pred):

    indices = np.where(y\_train\_pred == cluster)

    mapped\_class = np.bincount(y\_train[indices]).argmax()

    cluster\_map[cluster] = mapped\_class

y\_train\_mapped = np.array([cluster\_map[c] for c in y\_train\_pred])

y\_test\_mapped = np.array([cluster\_map[c] for c in y\_test\_pred])

# -----------------------------

# 6. Compute accuracy

# -----------------------------

train\_acc = accuracy\_score(y\_train, y\_train\_mapped)

test\_acc = accuracy\_score(y\_test, y\_test\_mapped)

print(f"Training Accuracy: {train\_acc:.2%}")

print(f"Test Accuracy: {test\_acc:.2%}")

# -----------------------------

# 7. Confusion Matrix Heatmap

# -----------------------------

cm = confusion\_matrix(y\_test, y\_test\_mapped)

plt.figure(figsize=(6, 5))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",

            xticklabels=target\_names, yticklabels=target\_names)

plt.xlabel("Predicted", fontsize=12)

plt.ylabel("Actual", fontsize=12)

plt.title("Confusion Matrix - Test Data", fontsize=14)

plt.show()