EXPERIMENT-10

CODE:-

```
# Expectation-Maximization Algorithm for Gaussian Mixture Model

import numpy as np
from scipy.stats import norm

# Initialize parameters
def initialize_parameters(X, k):
    np.random.seed(0)
    means = np.random.choice(X.flatten(), k)
    variances = np.random.rand(k)
    weights = np.ones(k) / k
    return means, variances, weights

# E-Step: Compute responsibilities
def expectation_step(X, means, variances, weights, k):
    responsibilities = np.zeros((len(X), k))

    for i in range(k):
        responsibilities[:, i] = weights[i] * norm.pdf(X, means[i], np.sqrt(variances[i]))

    responsibilities = responsibilities / responsibilities.sum(axis=1, keepdims=True)
    return responsibilities

# M-Step: Update parameters
def maximization_step(X, responsibilities, k):
    n = len(X)
```

```
    means = np.zeros(k)
    variances = np.zeros(k)
    weights = np.zeros(k)

    for i in range(k):
        resp_sum = responsibilities[:, i].sum()
        means[i] = np.sum(responsibilities[:, i] * X) / resp_sum
        variances[i] = np.sum(responsibilities[:, i] * (X - means[i])**2) / resp_sum
        weights[i] = resp_sum / n

    return means, variances, weights

# EM Algorithm
def em_algorithm(X, k, iterations=100):
    means, variances, weights = initialize_parameters(X, k)

    for _ in range(iterations):
        responsibilities = expectation_step(X, means, variances, weights, k)
        means, variances, weights = maximization_step(X, responsibilities, k)

    return means, variances, weights

# Sample dataset
X = np.array([1.2, 1.8, 2.5, 3.0, 3.6, 4.2, 4.8, 5.3, 6.0])

# Number of clusters
k = 2

# Run EM Algorithm
```

```
means, variances, weights = em_algorithm(X, k)

# Display results
print("Final Means:", means)
print("Final Variances:", variances)
print("Final Weights:", weights)
```

OUTPUT:-

```
Final Means: [4.86738851 2.3223678 ]
Final Variances: [0.70767038 0.72499773]
Final Weights: [0.5020125 0.4979875]
```