7	Aim	Implement and demonstrate the working model of K-means clustering algorithm with Expectation Maximization Concept
	Program	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 these two algorithms and comment on the quality of clustering. You can add Python ML library classes/API in the program

CONCEPT - Expectation-Maximization (EM)

- Clustering is a type of unsupervised learning wherein data points are grouped into different sets based on their degree of similarity.
- The Expectation-Maximization (EM) algorithm is a popular method for finding maximum likelihood estimates of parameters in statistical models, particularly when the model depends on unobserved latent variables. In the context of clustering, the EM algorithm is often used with Gaussian Mixture Models (GMMs).
- The goal of clustering is to partition data points into clusters such that points within the same cluster are more similar to each other than to points in other clusters. GMMs assume that the data points are generated from a mixture of several Gaussian distributions, each representing a cluster.

Steps of the EM Algorithm

1. Initialization: Initialize the parameters of the Gaussian components such as means, covariances, and mixing coefficients. These can be initialized randomly or using some heuristic method.

2. Expectation Step (E-Step):

   - Calculate the responsibility that each Gaussian component takes for each data point. This is done using the current parameter estimates.
   - The responsibility r_{ij} is the probability that the j-th data point belongs to the i-th Gaussian component.

$$r_{ij} = \frac{\pi_i \mathcal{N}(x_j | \mu_i, \Sigma_i)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_j | \mu_k, \Sigma_k)}$$

where π_i is the mixing coefficient, $\mathcal{N}(x_j | \mu_i, \Sigma_i)$ is the Gaussian probability density function, and K is the number of Gaussian components.

3. **Maximization Step (M-Step):**

- Update the parameters of the Gaussian components using the responsibilities calculated in the E-Step.
- The new parameters are calculated as follows:
    - **Mixing Coefficients:**

$$\pi_i = \frac{1}{N} \sum_{j=1}^{N} r_{ij}$$

- **Means:**

$$\mu_i = \frac{\sum_{j=1}^{N} r_{ij} x_j}{\sum_{j=1}^{N} r_{ij}}$$

- **Covariances:**

$$\Sigma_i = \frac{\sum_{j=1}^{N} r_{ij} (x_j - \mu_i)(x_j - \mu_i)^T}{\sum_{j=1}^{N} r_{ij}}$$

where N is the number of data points.

4. **Convergence Check:**

- Check if the parameters have converged. This can be done by monitoring the change in the log-likelihood of the data given the model parameters. If the change is below a certain threshold, the algorithm has converged.

$$\log L = \sum_{j=1}^{N} \log \left(\sum_{i=1}^{K} \pi_i \mathcal{N}(x_j | \mu_i, \Sigma_i) \right)$$

5. **Repeat:**

- Repeat the E-Step and M-Step until convergence.

K – Means Clustering

The algorithm will categorize the items into k groups of similarity. To calculate that similarity, use the Euclidean distance as measurement.

Algorithm

```
Input: Data points X = {x1, x2, ..., xn}, number of clusters K
Output: Cluster assignments for each data point

1. Initialize K cluster centroids randomly
2. Repeat until convergence or a maximum number of iterations:
        a. Assign each data point $x_i$ to the nearest centroid
        b. Recalculate the centroids as the mean of the data points assigned to each cluster
3. Return the cluster assignments
```

silhouette_score

The **silhouette_score** is a metric used to evaluate the quality of clustering results. It measures how similar an object is to its own cluster compared to other clusters. The silhouette score ranges from -1 to 1, where:

- **1** indicates that the data points are well-clustered, meaning they are close to other points in the same cluster and far from points in other clusters.
- **0** indicates that the data points are on or very close to the decision boundary between two neighboring clusters.
- **-1** indicates that the data points may have been assigned to the wrong cluster, as they are closer to points in other clusters than to points in their own cluster.

Formula

$$s = \frac{b-a}{\max(a,b)}$$

The silhouette score for a single data point is calculated using the formula:

- a - is the mean distance between a data point and all other points in the same cluster.
- b - is the mean distance between a data point and all other points in the nearest cluster (i.e., the cluster that minimizes this mean distance).

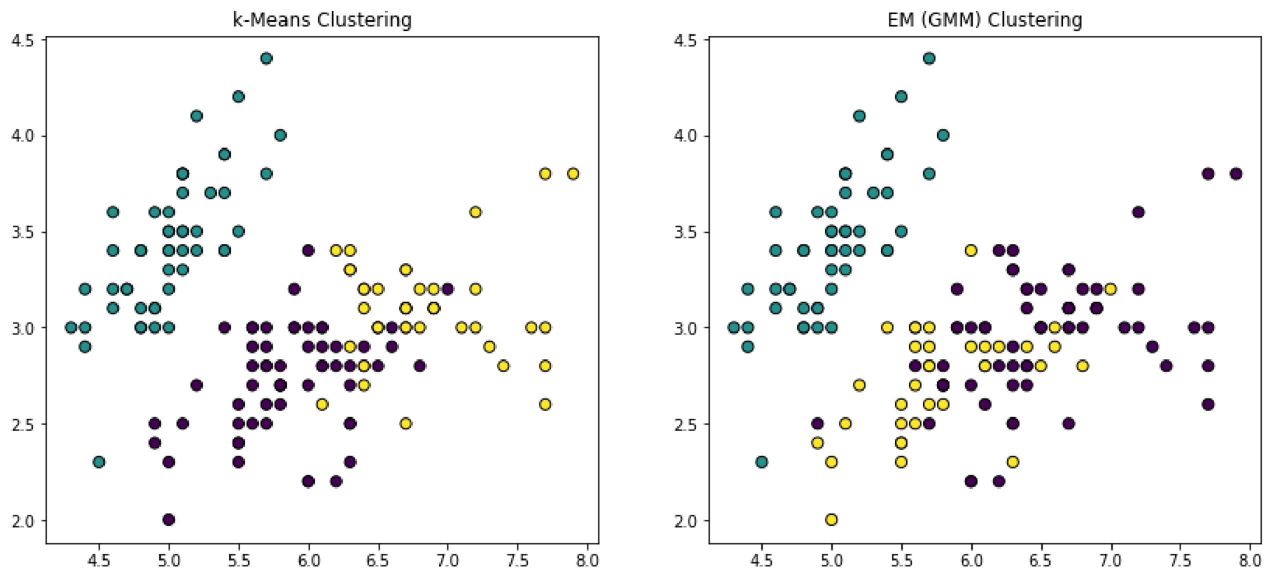
The overall silhouette score for a clustering solution is the mean silhouette score of all data points

Program:

```
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score

# Step 1: Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Step 2: Cluster the data using k-Means algorithm
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(X)
kmeans_silhouette = silhouette_score(X, kmeans_labels)

# Step 3: Cluster the data using EM algorithm (Gaussian Mixture Model)
gmm = GaussianMixture(n_components=3, random_state=42)
gmm_labels = gmm.fit_predict(X)
gmm_silhouette = silhouette_score(X, gmm_labels)

# Step 4: Compare the clustering results and visualize them
print(f"Silhouette Score for k-Means: {kmeans_silhouette}")
print(f"Silhouette Score for EM (GMM): {gmm_silhouette}")

# Visualization of the clusters
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

# k-Means Clustering
ax1.scatter(X[:, 0], X[:, 1], c=kmeans_labels, cmap='viridis', marker='o', edgecolor='k',
s=50)
ax1.set_title('k-Means Clustering')

# EM (GMM) Clustering
ax2.scatter(X[:, 0], X[:, 1], c=gmm_labels, cmap='viridis', marker='o', edgecolor='k',
s=50)
ax2.set_title('EM (GMM) Clustering')

plt.show()
```

```
# Step 5: Comment on the quality of clustering
if kmeans_silhouette > gmm_silhouette:
    print("k-Means clustering provides better quality clusters according to the silhouette score.")
else:
    print("EM (GMM) clustering provides better quality clusters according to the silhouette score.")
```

Output



```
Silhouette Score for k-Means: 0.551191604619592
Silhouette Score for EM (GMM): 0.5011761635067206

k-Means clustering provides better quality clusters according to
the silhouette score.
```