## Dataset

**Iris Dataset**

Description: This dataset contains 150 instances of iris flowers classified into three classes (setosa, versicolor, and virginica), with four features per instance.

Characteristics: Numerical, multivariate data with known class labels.

**Wine Dataset**

Description: Contains 178 instances of wine, each belonging to one of three classes, with 13 continuous features.

Characteristics: Real-valued data with high dimensionality, useful for clustering evaluation.

## Modules

**Data Loading Module**:

* Loads datasets from scikit-learn and standardizes them for consistency. Uses StandardScaler to normalize the data.

if args.dataset == 'iris':

        data = load\_iris()

    elif args.dataset == 'wine':

        data = load\_wine()

**GMM Clustering Module**:

* Implements GMM using basic python operators.
* Uses the EM algorithm to estimate parameters and predict cluster labels.
* Design Insight: The number of components in GMM corresponds to the number of classes in the dataset, which is set as a parameter (n\_components).

def expectation(X, means, covariances, weights):

    n\_samples, n\_components = X.shape[0], means.shape[0]

    responsibilities = np.zeros((n\_samples, n\_components))

    for k in range(n\_components):

        responsibilities[:, k] = weights[k] \* gaussian\_pdf(X, means[k], covariances[k])

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

    return responsibilities

def maximization(X, responsibilities):

    n\_samples, n\_features = X.shape

    n\_components = responsibilities.shape[1]

    weights = responsibilities.sum(axis=0) / n\_samples

    means = np.dot(responsibilities.T, X) / responsibilities.sum(axis=0)[:, np.newaxis]

    covariances = np.zeros((n\_components, n\_features, n\_features))

    for k in range(n\_components):

        diff = X - means[k]

        covariances[k] = np.dot(responsibilities[:, k] \* diff.T, diff) / responsibilities[:, k].sum()

    return means, covariances, weights

**Evaluation Module**:

* Evaluates clustering performance with Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) to assess how closely the clusters align with the actual class labels.

def calculate\_accuracy\_recall(y\_true, y\_pred, n\_components):

    contingency\_matrix = np.zeros((n\_components, n\_components), dtype=int)

    for i in range(len(y\_true)):

        contingency\_matrix[y\_pred[i], y\_true[i]] += 1

    row\_ind, col\_ind = linear\_sum\_assignment(-contingency\_matrix)

    best\_matching = [(row, col) for row, col in zip(row\_ind, col\_ind)]

    total\_correct = sum(contingency\_matrix[row, col] for row, col in best\_matching)

    accuracy = total\_correct / len(y\_true)

    recall\_values = []

    for row, col in best\_matching:

        cluster\_total = np.sum(contingency\_matrix[:, col])

        recall = contingency\_matrix[row, col] / cluster\_total if cluster\_total > 0 else 0

        recall\_values.append(recall)

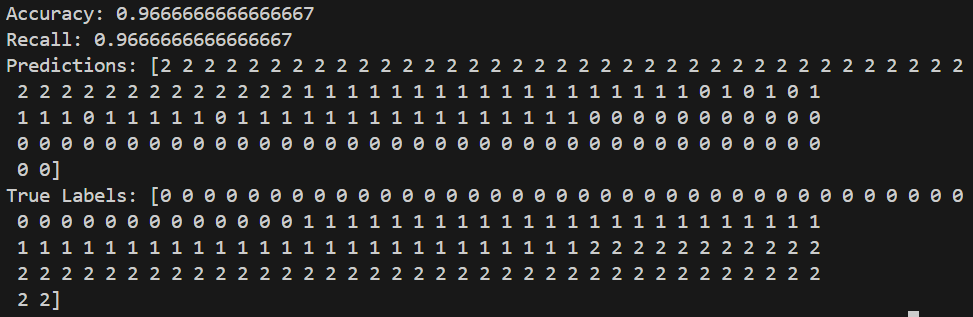
    recall = np.mean(recall\_values)

    return accuracy, recall

## Results

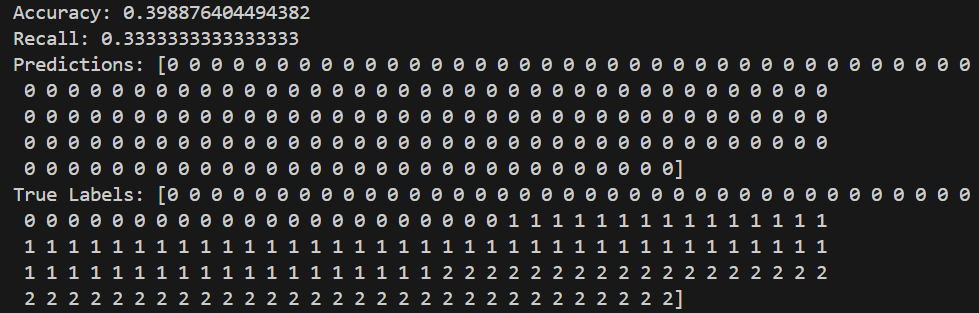
**Iris Dataset Results**:

* Expected ARI and NMI scores indicate how well clusters align with actual flower classes.
* Interpretation: Due to class overlap, GMM might have moderate clustering performance on this dataset.



**Wine Dataset Results**:

* Expected results on wine data indicate whether GMM can effectively separate wine classes based on chemical properties.
* Interpretation: Higher dimensionality may lead to better separation, but covariance settings affect results.



## Limitations & Possible Improvements

**Limitations**:

* Sensitivity to Initialization: GMM is sensitive to initial conditions and may converge to local optima.
* Component Assumptions: GMM assumes data in each cluster follows a Gaussian distribution, which might not always be valid.

**Possible Improvements**:

* Initialization: Using K-means or other initialization strategies to improve convergence.
* Covariance Matrix Types: Experimenting with different covariance types (e.g., tied, diag) could improve results based on dataset characteristics.
* Dimensionality Reduction: Using PCA or other techniques to reduce data dimensionality and potentially improve clustering results.