1. The Maximum Likelihood Estimation (MLE) for the parameters of a Gaussian Mixture Model (GMM) involves 2 steps: the E-step (Expectation Step) and the M-step (Maximization Step).

Maximisation step:

- Given the posterior probability $\gamma_k^{(n)} = p(z^{(n)} = k \mid x^{(n)})$, we want to update the model parameters $\theta = \{\mu, \Sigma, \pi\}$ by maximising the expected log likelihood, i.e.,

$$\max_{\theta} \sum_{n}^{K} \sum_{k}^{K} \gamma_k^{(n)} \ln(\pi_k) + \sum_{n}^{K} \sum_{k}^{K} \gamma_k^{(n)} \ln (\mathcal{N}(x^{(n)} \mid \mu_k, \Sigma_k)), \quad \text{s.t. } \sum_{k}^{K} \pi_k = 1$$

Summary of EM: Initialize the means μ_k, covariances Σ_k and mixing coefficients π_k

Iterate until convergence:

- E-step: Evaluate the responsibilities given current parameters

$$\gamma_k^{(n)} = p(z^{(n)} = k \mid x^{(n)}; \theta) = \frac{\pi_k \mathcal{N}(x^{(n)} \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x^{(n)} \mid \mu_j, \Sigma_j)}$$

- M-step: Re-estimate the parameters given current responsibilities

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} x^{(n)}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} (x^{(n)} - \mu_k)(x^{(n)} - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N}, \text{ with } N_k = \sum_{n=1}^{N} \gamma_k^{(n)}$$

- Assuming the data was really generated this way, update the parameters of each Gaussian component to maximize probability that it would generate the data it is currently responsible for.
  - ↳ Each Gaussian get a certain amount of posterior probability for each datapoint.
  - ↳ We fit each Gaussian to the weighted datapoints
  - ↳ We can derive closed form updates for all parameters

- Evaluate loglikelihood and check for convergence

$$\ln p(X \mid \pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x^{(n)} \mid \mu_k, \Sigma_k) \right)$$

2. Predictions:

↳ Prediction is a measure of the accuracy of the positive predictions. It is calculated as the ratio of true positive predictions to the total number of positive predictions, expressed mathematically as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall (Sensitivity or True Positive Rate):

↳ Recall measures the ability of the classifier to capture all the positive instances. It is calculated as the ratio of true positive predictions to the total number of ~~active~~ actual positives, expressed mathematically as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

ROC Curve (Receiver Operating Characteristic):

↳ The ROC curve is a graphical representation of the trade-off between sensitivity (TPR) and specificity (TNR) for different threshold values. It plots the true positive rate against the false positive rate. The curve is created by varying the decision threshold of the classifier, and points on the curve represents different trade-offs between sensitivity and specificity.

AUC (Area under curve)

↳ AUC quantifies the overall performance of a classification model by calculating the area under the ROC curve. A higher AUC indicates better discrimination between positive and negative classes. A perfect classifier has an AUC of 1, while a random classifier has an AUC of 0.5.

The AUC is calculated by integrating the ROC curve. It represents the probability that a randomly chosen positive instance will be ranked higher than a randomly chosen negative instance.

AUC is a useful metric when evaluating the performance of a classifier across various threshold values.

In a highly imbalanced dataset where the negative class dominates, the ROC curve may still provide a reasonable representation of the classifier's performance because it focuses on the trade-off between true positive and false positive rates. The false positive rates involves the true negative count, which is usually high in imbalanced datasets.

On the other hand, the PR curve, which involves Precision (which is sensitive to false positives), may be heavily influenced by the imbalance. In situations where there are many more negative samples than positive samples, false positives can have a significant impact on precision, leading to a potentially misleading representation of classifier's performance.

In summary, the PR curve is generally more affected by class imbalance than the ROC curve.

ROC curve:

- The ROC curve plots the TPR (sensitivity) against the FPR (1-specificity) for different threshold values.
- Mathematically, TPR is given by $TPR = \frac{TP}{TP+FN}$, and FPR is given by $FPR = \frac{FP}{FP+TN}$
- The ROC curve is less affected by the imbalance in class distribution because both the numerator and denominator of TPR and FPR involve the same class (either positive or negative)

PR curve:

- The PR curve plots Precision against Recall for different threshold values.
- Mathematically, Precision is given by $Precision = \frac{TP}{TP+FP}$ and Recall is given by $Recall = \frac{TP}{TP+FN}$
- Precision is more sensitive to class imbalance because it depends on the number of FP, which can be high in the case of an uneven distribution of positive and negative samples.

3. (20 points) Consider the following 10 data points: $X = \{(1,0,2,-3,-2),(0,1,-3,-2,-3),(1,2,1,3,-2),(-1,1,2,3,-1),(1,0,1,-1,1),(2,3,-1,1,-2),(-2,3,-3,2,3),(-2,-2,2,3,-2),(-2,-2,1,-3,-3),(-3,2,0,-1,-2)\}$. Compute the unit-length principle components of X and choose two of them for PCA, then calculate the projection of each data on these two principal components. You could use python or matlab to obtain eigenvectors and eigenvalues.
end

```
[2] X = np.matrix([[1,0,2,-3,-2], [0,1,-3,-2,-3], [1,2,1,3,-2], [-1,1,2,3,-1], [1,0,1,-1,1],
                   [2,3,-1,1,-2], [-2,3,-3,2,3], [-2,-2,2,3,-2], [-2,-2,1,-3,-3], [-3,2,0,-1,-2]])
    X = X.T
    print(X)
    print('Length Data:', len(X.T))

    [[ 1  0  1 -1  1  2 -2 -2 -2 -3]
     [ 0  1  2  1  0  3  3 -2 -2  2]
     [ 2 -3  1  2  1 -1 -3  2  1  0]
     [-3 -2  3  3 -1  1  2  3 -3 -1]
     [-2 -3 -2 -1  1 -2  3 -2 -3 -2]]
    Length Data: 10
```

```
X_mean = np.mean(X, axis = 1)
X = X - X_mean
X

matrix([[ 1.5,  0.5,  1.5, -0.5,  1.5,  2.5, -1.5, -1.5, -1.5, -2.5],
        [-0.8,  0.2,  1.2,  0.2, -0.8,  2.2,  2.2, -2.8, -2.8,  1.2],
        [ 1.8, -3.2,  0.8,  1.8,  0.8, -1.2, -3.2,  1.8,  0.8, -0.2],
        [-3.2, -2.2,  2.8,  2.8, -1.2,  0.8,  1.8,  2.8, -3.2, -1.2],
        [-0.7, -1.7, -0.7,  0.3,  2.3, -0.7,  4.3, -0.7, -1.7, -0.7]])
```

```
[4] cov = np.dot(X, X.T) / len(X.T)
    cov

    matrix([[ 2.65,  0.7 ,  0.1 , -0.2 , -0.25],
            [ 0.7 ,  2.96, -1.86,  1.24,  1.14],
            [ 0.1 , -1.86,  3.36,  0.36, -0.94],
            [-0.2 ,  1.24,  0.36,  5.56,  1.36],
            [-0.25,  1.14, -0.94,  1.36,  3.21]])
```

```
w, v = np.linalg.eig(cov)
print('Eigenvalues: \n', w)
print('Normalized Eigenvectors: \n', v)

Eigenvalues:
 [0.77040329 1.98945767 2.97959483 4.90948779 7.09105643]
Normalized Eigenvectors:
 [[-0.32834028  0.35316831  0.86657302  0.12816499  0.00947335]
 [ 0.72524475 -0.17268194  0.28300348  0.38603495  0.46377479]
 [ 0.55670647  0.36038392  0.16695994 -0.67533319 -0.27613549]
 [-0.23575903 -0.2628404   0.09886262 -0.60066664  0.71046349]
 [-0.02661412  0.80404692 -0.36236831  0.13294897  0.45145766]]
```

Two principal components with biggest values of λ

Unit length principal components of X represented by

$$U = \begin{pmatrix} 0.1282 & 0.0095 \\ 0.3860 & 0.4638 \\ -0.6753 & -0.2761 \\ -0.6007 & 0.7105 \\ 0.1329 & 0.4515 \end{pmatrix}$$

Result Projection: $U^T X$

Projection of each data on the principal components = $U^T X$

```
[6] U = v[:,3:]
    U

    matrix([[ 0.12816499,  0.00947335],
            [ 0.38603495,  0.46377479],
            [-0.67533319, -0.27613549],
            [-0.60066664,  0.71046349],
            [ 0.13294897,  0.45145766]])
```

Unit length principal components of X

```
#projection
pca_x = np.dot(U.T, X)
pca_x.T

matrix([[ 0.49688877, -3.44335723],
        [ 3.39780904, -1.34937249],
        [-1.65970801,  2.0231088 ],
        [-2.84445715,  1.71570947],
        [ 0.36973558, -0.39192178],
        [ 1.40649158,  1.62770094],
        [ 2.30857621,  5.10983029],
        [-4.26367594, -0.1365459 ],
        [-0.11729188, -4.57464902],
        [ 0.90563179, -0.58050309]])
```

Point 1 on 2D PCA
Point 2 on 2D PCA
.
.
.
.
.
Point 10 on 2D PCA

Projection result in 2D principal component