# Posterior Probability

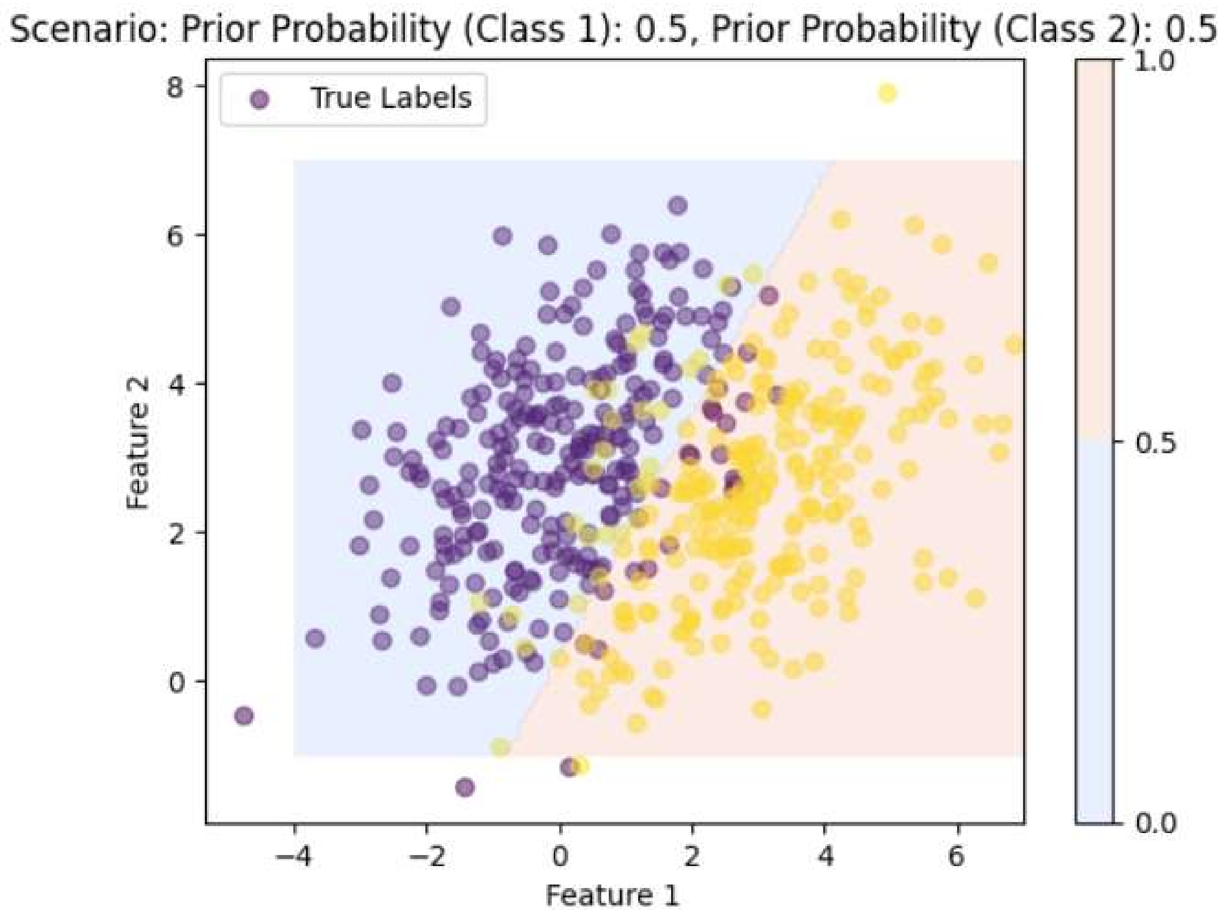
- probabilities assigned to each class after observing the features of a particular data point.
- help in decision-making by providing a measure of confidence or uncertainty in the classification of a data point into a particular class.

$$P(C_i|X) = \frac{P(X|C_i) \cdot P(C_i)}{P(X)}$$

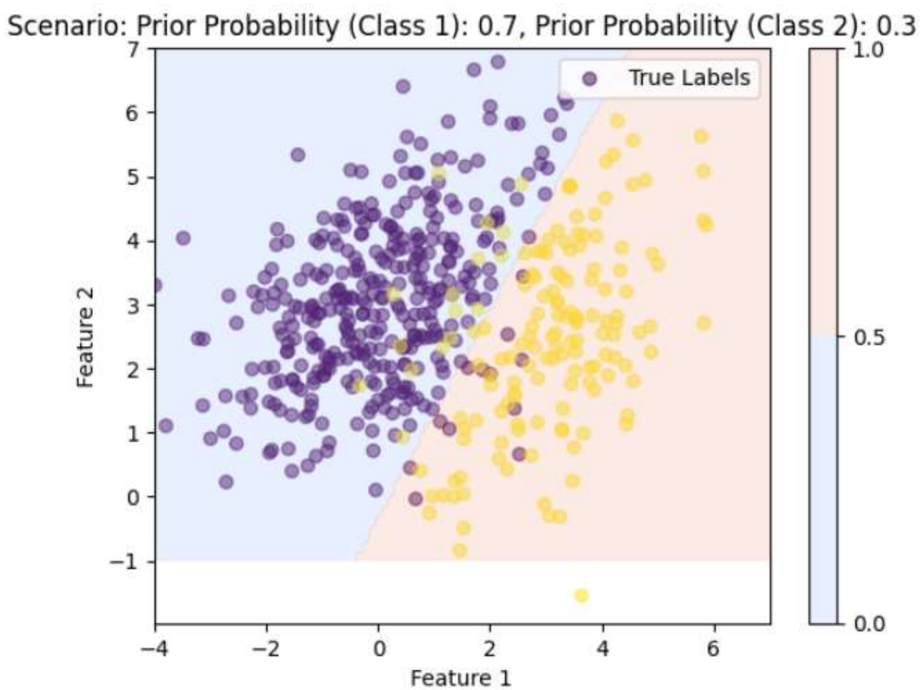
where:

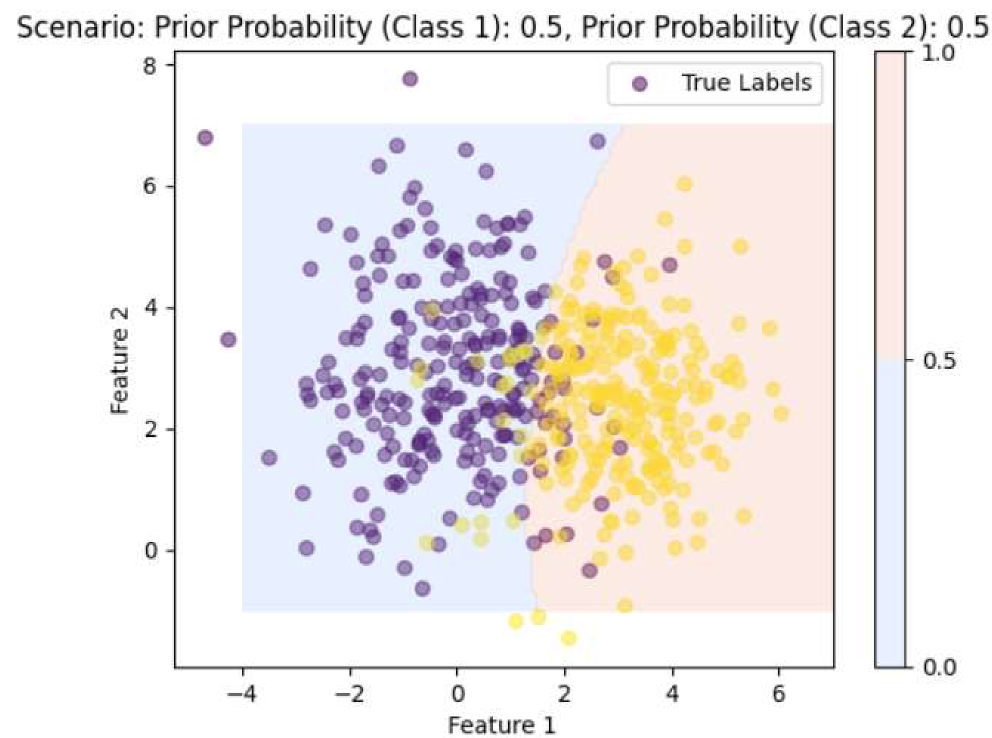
- $P(C_i|X)$ is the posterior probability of class $C_i$ given the observed features $X$.
- $P(X|C_i)$ is the likelihood of observing features $X$ given class $C_i$.
- $P(C_i)$ is the prior probability of class $C_i$.
- $P(X)$ is the probability of observing the features $X$ across all classes (also known as the evidence).

# Posterior probability for classification



# Posterior probability for classification



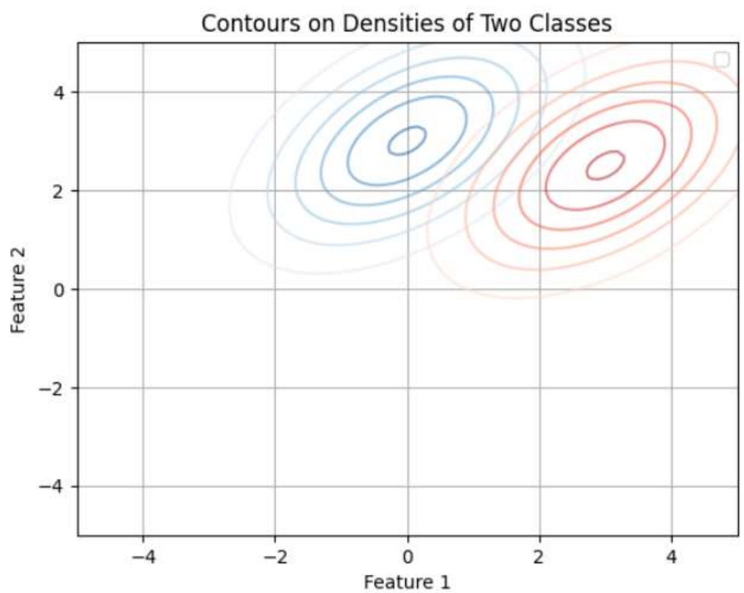



# Fisher Linear Discriminant Analysis

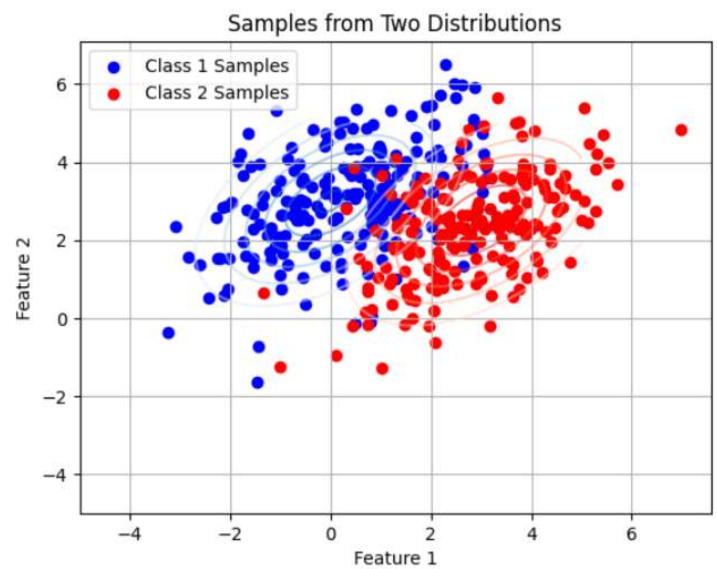
- to find a linear combination of features that best separates different classes in a dataset.

**Considering the example given in lab sheet**

1.




2.

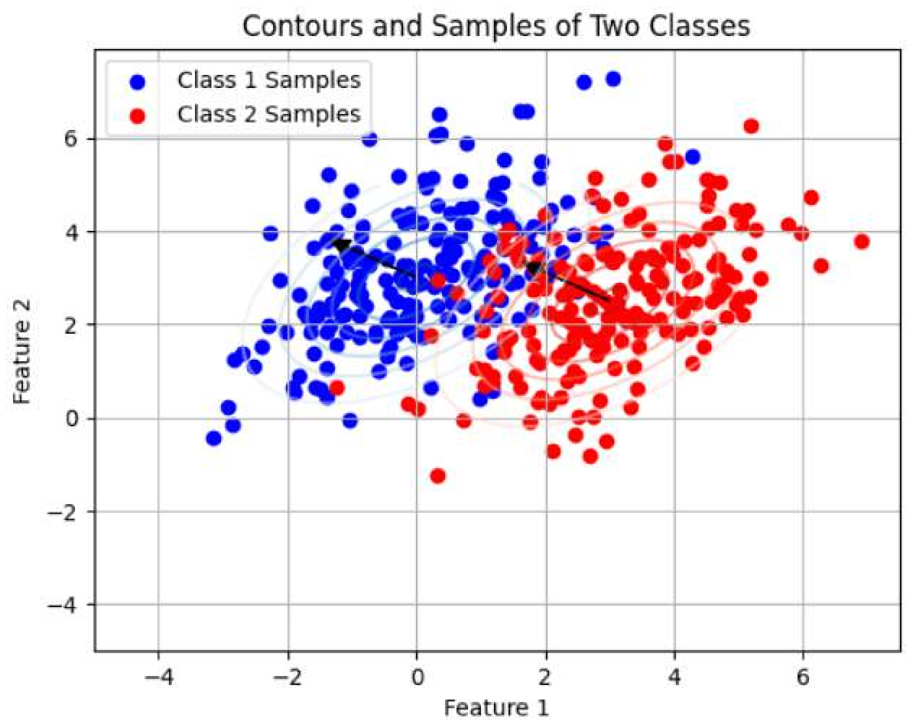


# Fisher Linear Discriminant Analysis

3. $w_F = (C_1 + C_2)^{-1}(m_1 - m_2)$

WF is the vector representing the direction of the linear discriminant.

- **wF[0] , wf[1]** represent the coefficient or weight of features in the Fisher Linear Discriminant direction.

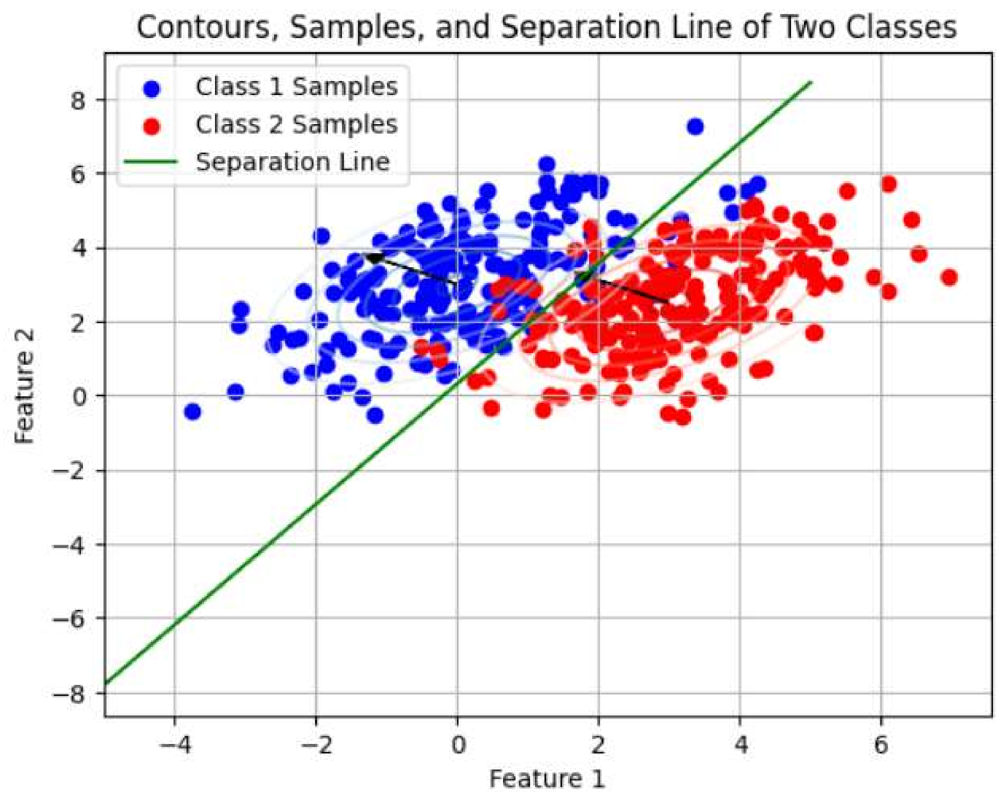


# Fisher Linear Discriminant Analysis

4. Draw the class boundary

$$m = -\frac{wF[0]}{wF[1]}$$

$$c = \frac{m1[1] + m2[1]}{2} - m \times \frac{m1[0] + m2[0]}{2}$$

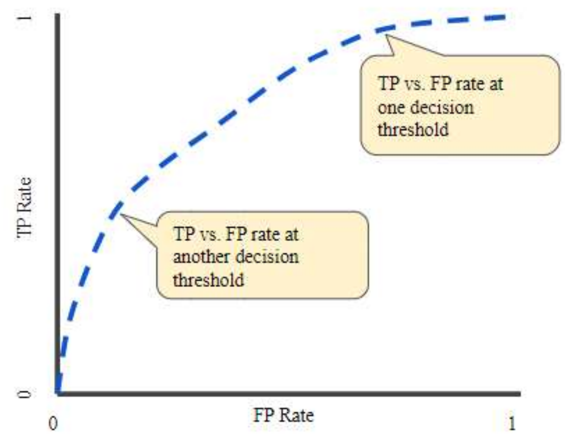


# ROC Curve

- Receiver Operating Characteristic (ROC) Curve
- fundamental tool for evaluating the performance of classification algorithms, especially in scenarios where the classes are imbalanced.
- ROC curve plots the true positive rate against the false positive rate at various threshold settings.

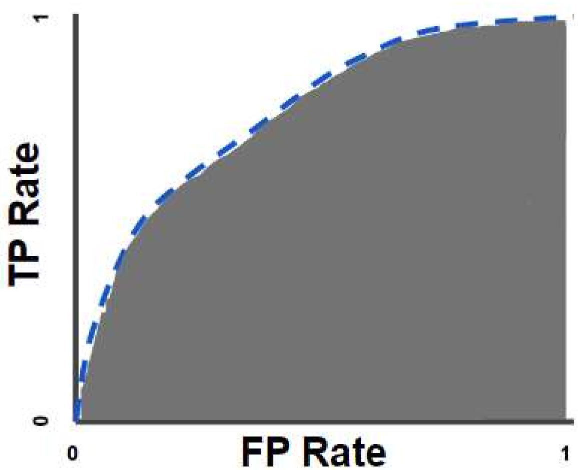
$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$



# Area Under the ROC Curve

- Area under the ROC curve quantifies the overall performance of the classifier




```python
import numpy as np

# Sample data (replace with your actual TPR and FPR values)
fpr = np.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
tpr = np.array([0.2, 0.4, 0.6, 0.7, 0.8, 0.9, 0.95, 0.98, 0.99, 1.0])

# Calculate AUC using trapezoidal integration
auc = np.trapz(tpr, fpr)

print("AUC:", auc)
```

# Mahalanobis Distance

- Mahalanobis distance is a measure used to determine the dissimilarity between two points in multivariate space.
  - **Mahalanobis distance-to-mean classifier**

  calculating the Mahalanobis distance of a new data point to the mean of each class using the respective class covariance matrix.

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{S}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

Where:

- $D_M(\mathbf{x})$ is the Mahalanobis distance between the point $\mathbf{x}$ and the distribution mean $\boldsymbol{\mu}$.
- $\mathbf{S}$ is the covariance matrix of the distribution.
- $\mathbf{S}^{-1}$ is the inverse of the covariance matrix.

# Mahalanobis Distance

- **Distance-to-mean classifier**

calculating the Euclidean distance of a new data point to the mean of each class.

$$d(\mathbf{x}, \boldsymbol{\mu}) = \sqrt{\sum_{i=1}^{n} (x_i - \mu_i)^2}$$

Where:

- $d(\mathbf{x}, \boldsymbol{\mu})$ is the Euclidean distance between the point $\mathbf{x}$ and the mean $\boldsymbol{\mu}$ of the class.
- $x_i$ and $\mu_i$ are the $i$th components of the data point and the mean vector respectively.
- $n$ is the number of features (dimensions) in the data.