Part A: Probabilistic Gaussian Generative Classifier

1. Model Explanation: We modeled the classification task using a Gaussian Generative Classifier (similar to Linear Discriminant Analysis). The model assumes the data is generated via the following probabilistic process:

- **Class Priors :p(y=k):** We assume the probability of observing any specific digit class k follows a categorical distribution. This represents the baseline frequency of each digit.
- **Class-Conditional Density p(x | y=k):** We assume the feature vectors (pixel intensities) for each class are distributed according to a Multivariate Gaussian distribution

  N=(Mk,∑)

-
  - Mk is the class-specific mean vector .
  - ∑ is a **shared covariance matrix**, meaning we assume all digit classes share the same shape and spread in the feature space, varying only by their location (mean).

2)Parameter Estimation:

The parameters were estimated using Maximum Likelihood Estimation on the training set:

- **Priors (pi_k):** Calculated as the fraction of training samples belonging to class.
- **Means (Mk):** Calculated as the arithmetic mean of all feature vectors belonging to class.
- **Shared Covariance (∑**): Estimated by computing the covariance of the centered data (subtracting the respective class mean from each sample) across the entire training set.

3)Covariance Regularization

We regularized the covariance matrix using the formula Sigma =Sigma + lambda.

This step is necessary because:

1. **Invertibility:** In high-dimensional spaces (64 features) with limited data, the empirical covariance matrix might be singular or ill-conditioned, making it impossible to invert (which is required for the Gaussian density calculation).
2. **Overfitting:** Adding lambda to the diagonal ("adding noise") smoothens the distribution. It prevents the model from relying too heavily on specific feature correlations that might be noise in the training data.

2. Hyperparameter Tuning (Validation Results)

The table below summarizes the effect of the regularization strength (λ) on model performance.

Lambda	Validation Accuracy
0.0001	**0.9444**
0.0010	**0.9444**
0.0100	**0.9444**
0.1000	**0.9444**

Best λ: 0.0001, Validation Accuracy: 0.9444

3. Final Test Results

The final model, retrained on the combined training and validation sets using $\lambda = 0.0001$, achieved the following performance on the hold-out test set:

- **Test Accuracy:** 96.30%
- **Macro Precision:** 0.9632
- **Macro Recall:** 0.9627
- **Macro F1-score:** 0.9625

Confusion Matrix:

[[27 0 0 0 0 0 0 0 0 0]

[0 26 0 0 0 0 0 0 1 1]

[0 0 26 0 0 0 0 0 0 0]

[0 0 0 28 0 0 0 0 0 0]

[0 0 0 0 27 0 0 0 0 0]

[0 0 0 0 0 27 0 0 0 0]

[0 0 0 0 0 0 27 0 0 0]

[0 0 0 0 0 0 0 27 0 0]

[0 3 0 0 0 0 0 0 22 1]

[0 1 0 0 0 1 0 1 1 23]]

4. My point of view:

- **Confusions between digits:**
    - Digit 1 is sometimes confused with 8 and 9.
    - Digit 8 is confused with 1 and 9.
    - Digit 9 is confused with 1, 5, 7, and 8.
- **Effect of λ on performance:**
    - All candidate λ values produced the same validation accuracy (0.9444), indicating the model is fairly robust to λ in this range.
    - Small λ (0.0001) was chosen as best, keeping Σ closest to the empirical covariance while avoiding numerical instability**.**
- **Strengths / weaknesses:**
    - Strength: High accuracy (0.963) and macro F1-score (0.9625) show good overall performance, and it requires no iterative training (unlike neural networks) and provides a clear probabilistic interpretation
    - Weakness: Misclassification occurs for visually similar digits (1, 8, 9), which is typical for a simple Gaussian generative model with a shared covariance and covariance assumption->By forcing all digits to share the same "shape" of variance, the model cannot capture class-specific variations. This limitation likely contributed to the misclassifications of the structurally more complex digits like 8 and 9.