📌 Analiza teoretyczna zadania

🔹 1. Klasyfikacja bayesowska: ogólny wstęp

Klasyfikacja bayesowska opiera się na regule Bayesa:

$$P(y = k \mid x) = \frac{P(x \mid y = k)P(y = k)}{P(x)}$$

gdzie:

- y to zmienna docelowa (klasa 0 lub 1),
- x to wektor cech,
- $P(x \mid y = k)$ to rozkład cech w klasie k,
- $P(y = k)$ to prawdopodobieństwo a priori klasy k.

Klasyfikator przypisuje nową obserwację do tej klasy, dla której $P(y = k \mid x)$ jest największe.

🔹 2. LDA — Linear Discriminant Analysis

📌 Założenia

- Cechy są warunkowo normalne:

$$X \mid y = k \sim \mathcal{N}(\mu_k, \Sigma)$$

  czyli **wspólna macierz kowariancji** Σ dla wszystkich klas.

- Oznacza to: brak interakcji pomiędzy cechami w kontekście rozróżniania klas.

📌 Wzory

- Estymatory:
  - Średnie klas:

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

  - Wspólna macierz kowariancji:

$$\hat{\Sigma} = \frac{1}{n-2} \left(\sum_{i:y_i=0} (x_i - \hat{\mu}_0)(x_i - \hat{\mu}_0)^T + \sum_{i:y_i=1} (x_i - \hat{\mu}_1)(x_i - \hat{\mu}_1)^T \right)$$

- Funkcja dyskryminacyjna (logit):

$$\delta(x) = \log\left(\frac{\pi_1}{\pi_0}\right) - \frac{1}{2}(\mu_1^T \Sigma^{-1} \mu_1 - \mu_0^T \Sigma^{-1} \mu_0) + x^T \Sigma^{-1} (\mu_1 - \mu_0)$$

- Prawdopodobieństwo klasy 1:

$$P(y = 1 \mid x) = \sigma(\delta(x)) = \frac{1}{1 + e^{-\delta(x)}}$$

📌 Implementacja

W klasie `LDA`:

- `fit(X, y)`: szacuje μ_0, μ_1 oraz Σ.
- `predict_proba(Xtest)`: oblicza $\delta(x)$ i używa funkcji sigmoidalnej (expit) do uzyskania prawdopodobieństwa.
- `predict(Xtest)`: przypisuje klasę 1, jeśli $P(y = 1 \mid x) > 0.5$.

🔹 3. QDA — Quadratic Discriminant Analysis

📌 Założenia

- Cechy są warunkowo normalne, ale **z osobnymi macierzami kowariancji**:

$$X \mid y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$$

📌 Wzory

- Estymatory:
  - Średnie jak w LDA.
  - Macierze kowariancji:

$$\hat{\Sigma}_k = \frac{1}{n_k - 1} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$$

- Funkcja dyskryminacyjna:

$$\delta(x) = \log\left(\frac{\pi_1}{\pi_0}\right) - \frac{1}{2}\log\left(\frac{|\Sigma_1|}{|\Sigma_0|}\right) - \frac{1}{2}\left[(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)\right]$$

- Prawdopodobieństwo klasy 1:

$$P(y=1 \mid x) = \sigma(\delta(x))$$

📌 Implementacja

W klasie `QDA`:

- `fit(X, y)`: estymuje μ_0, μ_1 i Σ_0, Σ_1.
- `predict_proba(Xtest)`: oblicza $\delta(x)$ i używa funkcji sigmoidalnej.

🔹 4. Naive Bayes (NB)

📌 Założenia

- Warunkowa niezależność cech:

$$P(x \mid y = k) = \prod_j P(x_j \mid y = k)$$

- Każda cecha jest modelowana osobno jako rozkład normalny:

$$x_j \mid y = k \sim \mathcal{N}(\mu_{kj}, \sigma_{kj}^2)$$

📌 Wzory

- Estymatory:

$$\hat{\mu}_{kj} = \frac{1}{n_k} \sum_{i:y_i=k} x_{ij}$$

$$\hat{\sigma}_{kj}^2 = \frac{1}{n_k} \sum_{i:y_i=k} (x_{ij} - \hat{\mu}_{kj})^2$$

- Log-likelihood:

$$\log P(x \mid y = k) = \sum_j -\frac{1}{2} \log(2\pi\sigma_{kj}^2) - \frac{(x_j - \mu_{kj})^2}{2\sigma_{kj}^2}$$

- Funkcja dyskryminacyjna:

$$\delta(x) = \log\left(\frac{\pi_1}{\pi_0}\right) + \log P(x \mid y = 1) - \log P(x \mid y = 0)$$

- Prawdopodobieństwo klasy 1:

$$P(y = 1 \mid x) = \sigma(\delta(x))$$

📌 Implementacja

W klasie `NB`:

- `fit(X, y)`: estymuje μ_{kj} i σ^2_{kj}.
- `predict_proba(Xtest)`: sumuje log-likelihoody.

🔹 5. Generowanie danych

📌 Scheme 1

- Klasa 0: cechy niezależne standardowe normalne.
- Klasa 1: cechy niezależne normalne z przesuniętą średnią a.

📌 Scheme 2

- Klasa 0: rozkład normalny 2D z korelacją ρ.
- Klasa 1: rozkład normalny 2D z korelacją $-\rho$.

🔹 6. Metryka jakości

- Accuracy:

$$Accuracy = \frac{\text{liczba poprawnych klasyfikacji}}{\text{liczba wszystkich obserwacji}}$$

📌 Podsumowanie

Twój kod:

✅ uwzględnia kluczowe założenia teoretyczne (np. wspólna/oddzielna macierz kowariancji w LDA/QDA, niezależność w NB).

✅ zawiera wzory dla obliczania dyskryminanty (logitu).

✅ implementuje wszystkie trzy metody w pełni od podstaw.

🔹 Użyte wzory i funkcje w kodzie

- Sigmoid (logit): `expit(x) = 1 / (1 + exp(-x))`
- Średnia: `np.mean()`
- Kowariancja: macierze przez `(X - mu).T @ (X - mu) / (n-1)`
- Log-determinanty i inwersje: `np.linalg.det()` i `np.linalg.inv()`
- Priory: `pi_k = n_k / n`