HW6_report

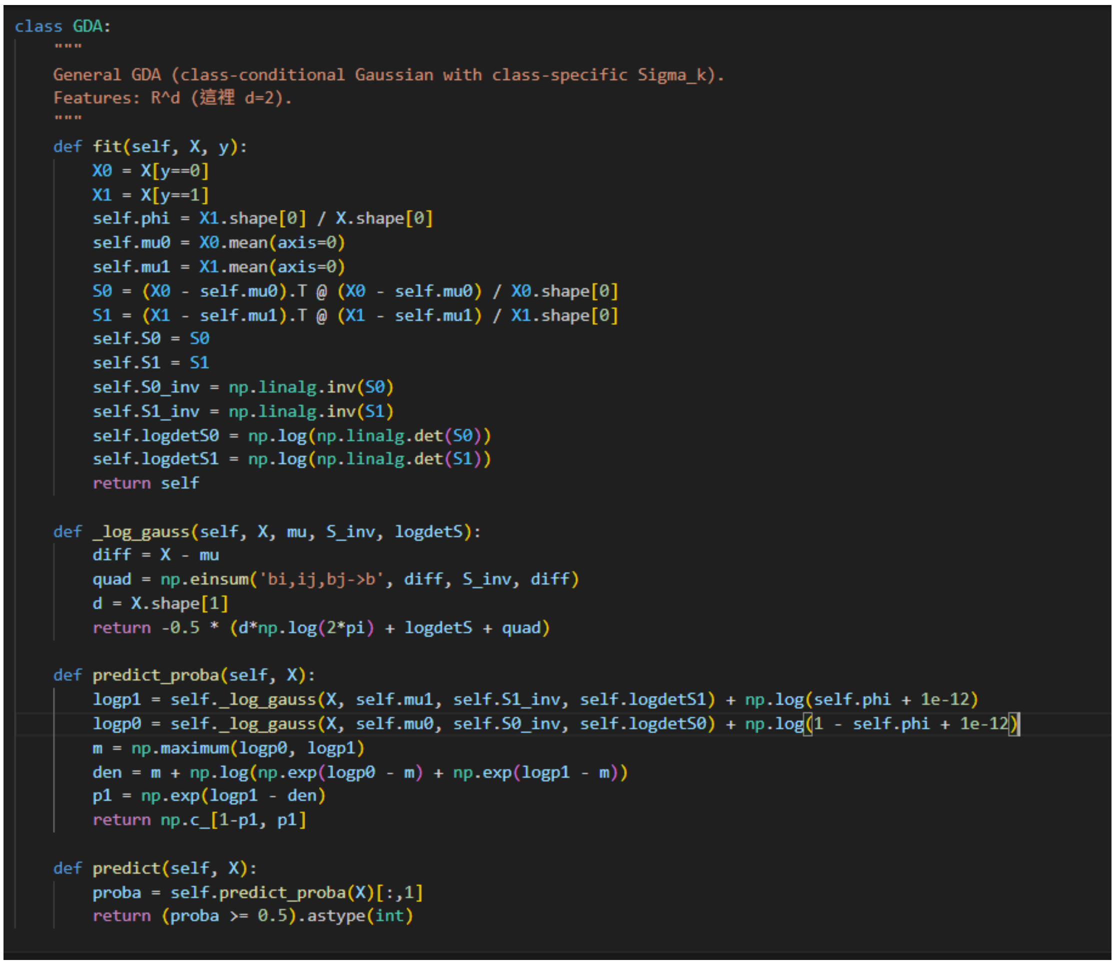
110652012 施品光

Part1

(a) I implemented the Gaussian Discriminant Analysis (GDA) model **from scratch**, without using any built-in classifier functions.

The model estimates the parameters $\mu_0, \mu_1, \Sigma_0, \Sigma_1$ and the class prior $\phi = P(y=1)$.

Predictions are made by comparing the log-posterior probabilities for both classes.



```python
class GDA:
    """
    General GDA (class-conditional Gaussian with class-specific Sigma_k).
    Features: R^d (這裡 d=2).
    """
    def fit(self, X, y):
        X0 = X[y==0]
        X1 = X[y==1]
        self.phi = X1.shape[0] / X.shape[0]
        self.mu0 = X0.mean(axis=0)
        self.mu1 = X1.mean(axis=0)
        S0 = (X0 - self.mu0).T @ (X0 - self.mu0) / X0.shape[0]
        S1 = (X1 - self.mu1).T @ (X1 - self.mu1) / X1.shape[0]
        self.S0 = S0
        self.S1 = S1
        self.S0_inv = np.linalg.inv(S0)
        self.S1_inv = np.linalg.inv(S1)
        self.logdetS0 = np.log(np.linalg.det(S0))
        self.logdetS1 = np.log(np.linalg.det(S1))
        return self

    def _log_gauss(self, X, mu, S_inv, logdetS):
        diff = X - mu
        quad = np.einsum('bi,ij,bj->b', diff, S_inv, diff)
        d = X.shape[1]
        return -0.5 * (d*np.log(2*pi) + logdetS + quad)

    def predict_proba(self, X):
        logp1 = self._log_gauss(X, self.mu1, self.S1_inv, self.logdetS1) + np.log(self.phi + 1e-12)
        logp0 = self._log_gauss(X, self.mu0, self.S0_inv, self.logdetS0) + np.log(1 - self.phi + 1e-12)
        m = np.maximum(logp0, logp1)
        den = m + np.log(np.exp(logp0 - m) + np.exp(logp1 - m))
        p1 = np.exp(logp1 - den)
        return np.c_[1-p1, p1]

    def predict(self, X):
        proba = self.predict_proba(X)[:,1]
        return (proba >= 0.5).astype(int)
```

(b) Gaussian Discriminant Analysis (GDA) is a **generative model** that assumes the data of each class follows a multivariate Gaussian distribution.

It estimates the parameters μ_k and Σ_k for each class k, and the prior $\phi =$

$P(y = 1)$.

During prediction, it computes: $\log p(x|y = k) + \log P(y = k)$ for both classes and assigns x to the one with the higher posterior probability.

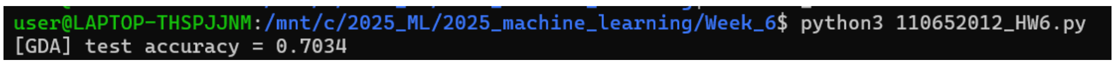
This dataset contains continuous features (longitude and latitude), making the Gaussian assumption reasonable.

GDA can model the spatial separation of valid and invalid points using a quadratic decision boundary.

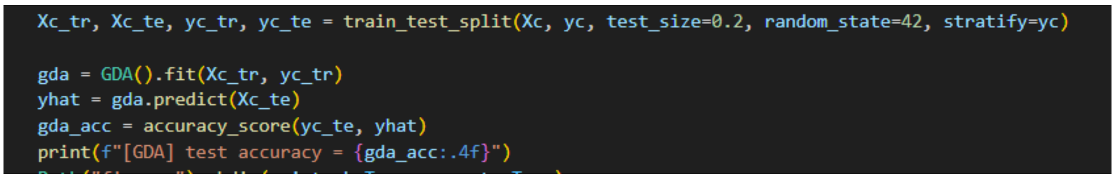
(c) The dataset was split into **80% training** and **20% testing** using train_test_split.

Model performance was evaluated by **accuracy on the test set**.

The model achieved:



```
user@LAPTOP-THSPJJNM:/mnt/c/2025_ML/2025_machine_learning/Week_6$ python3 110652012_HW6.py
[GDA] test accuracy = 0.7034
```



```
    Xc_tr, Xc_te, yc_tr, yc_te = train_test_split(Xc, yc, test_size=0.2, random_state=42, stratify=yc)

    gda = GDA().fit(Xc_tr, yc_tr)
    yhat = gda.predict(Xc_te)
    gda_acc = accuracy_score(yc_te, yhat)
    print(f"[GDA] test accuracy = {gda_acc:.4f}")
```

(d) The decision boundary of the GDA classifier is shown in Figure 1.

The yellow region represents samples classified as valid (1), while the blue region represents invalid (0).

Dots represent the actual data points.

The boundary appears mostly vertical, suggesting that longitude contributes more strongly to classification.

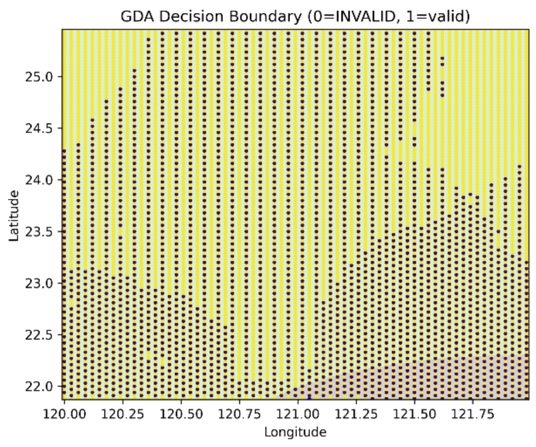
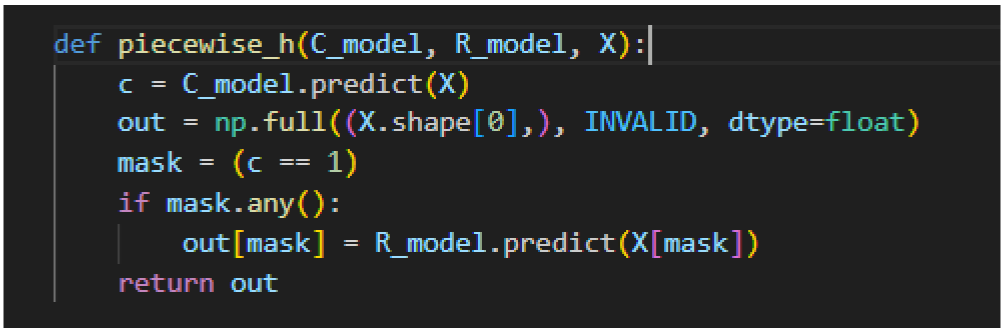


Figure 1. GDA Decision Boundary:

Part2

(a) I reuse the Week-4 models: a Random Forest classifier $C(\mathbf{x})$and a KNN regressor $R(\mathbf{x})$.

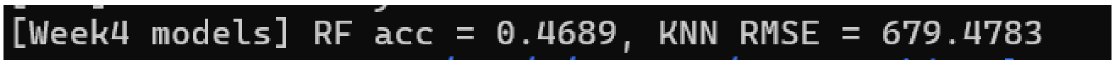
The combined function is defined as $h(\mathbf{x}) = R(\mathbf{x})$if $C(\mathbf{x}) = 1$, and -999 otherwise.



```
def piecewise_h(C_model, R_model, X):
    c = C_model.predict(X)
    out = np.full((X.shape[0],), INVALID, dtype=float)
    mask = (c == 1)
    if mask.any():
        out[mask] = R_model.predict(X[mask])
    return out
```

(b) I train $C(\mathbf{x})$on the classification set and $R(\mathbf{x})$on the valid-only regression set, then evaluate on held-out splits.

The results are:



```
[Week4 models] RF acc = 0.4689, KNN RMSE = 679.4783
```

(c) The combined function first filters inputs by the classifier: $C(\mathbf{x}) \in \{0,1\}$.

For samples predicted as valid ($C(\mathbf{x}) = 1$), it passes the same features to the regressor $R(\mathbf{x})$to obtain a continuous value.

For samples predicted invalid, it returns the sentinel value -999.

This design enforces task logic: prediction is meaningful only where the location is considered valid.

(d) Most grid points are purple, indicating $-999$where the classifier predicts invalid.

Small clusters of yellow/green appear where $C(\mathbf{x}) = 1$, showing the regressed values from $R(\mathbf{x})$.

This confirms the combined function behaves exactly as defined.

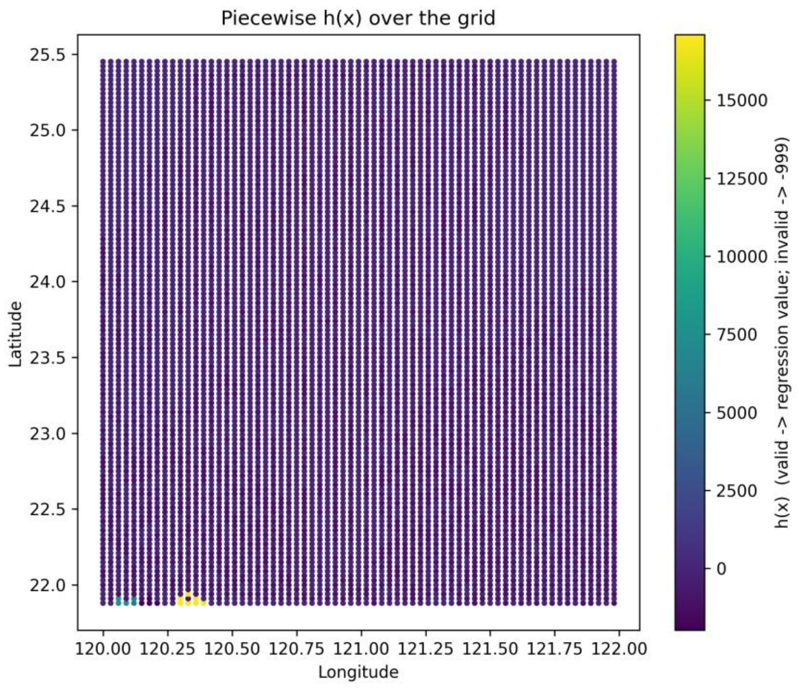


Figure 2:

Writing assignment

What are the limitations of GDA when the true data distribution is not Gaussian?

- For example, if the classes have skewed or multimodal distributions, how badly does the GDA assumption break?