Programming assignment

1. Classification

A. Write your own code to implement the GDA algorithm

```python
class GDA:
    def __init__(self):
        self.phi = None
        self.mu_0 = None
        self.mu_1 = None
        self.sigma = None
        self.sigma_inv = None

    def fit(self, X, y):
        n_samples, _ = X.shape
        # 1. 計算類別 1 的先驗機率 Φ
        self.phi = np.mean(y == 1)
        # 2. 計算兩個類別的特徵平均值 μ₀, μ₁
        self.mu_0 = np.mean(X[y == 0], axis=0)
        self.mu_1 = np.mean(X[y == 1], axis=0)
        # 3. 計算共享的共變異數矩陣
        n_0, n_1 = np.sum(y == 0), np.sum(y == 1)
        cov_0 = np.cov(X[y == 0].T, bias=True)
        cov_1 = np.cov(X[y == 1].T, bias=True)
        self.sigma = (n_0 * cov_0 + n_1 * cov_1) / n_samples
        self.sigma_inv = np.linalg.inv(self.sigma)

    def predict(self, X):
        term_0 = np.sum((X - self.mu_0) @ self.sigma_inv * (X - self.mu_0), axis=1)
        term_1 = np.sum((X - self.mu_1) @ self.sigma_inv * (X - self.mu_1), axis=1)
        log_posterior_0 = np.log(1 - self.phi) - 0.5 * term_0
        log_posterior_1 = np.log(self.phi) - 0.5 * term_1
        return (log_posterior_1 > log_posterior_0).astype(int)
```

依序計算了有效值的ϕ和兩個 label 的μ_1, μ_0以及共變異矩陣

B. Clearly explain how the GDA model works and why it can be used for classification, in particular this data set.

GDA 假設每個 label 的資料 p(x|y) 都服從多變數常態分佈。
透過學習（μ_1, μ_0和共變異數 ），結合先驗機率 p(y)，最終使用貝氏定理來預測新資料點。

因為資料室地理座標(群聚)，因此可以使用 GDA。

C. Train your model on the given dataset and report its accuracy. Be explicit about how you measure performance.

使用 Accuracy 作為評斷指標，GDA 模型在測試集上的準確率:
51.43%

D. Plot the decision boundary of your model and include the visualization in your report.

2. Regression

A. Implement this combined model in code.

```
class PiecewiseRegressionModel:
    def __init__(self, classification_model, regression_model, feature_mu, feature_sigma, label_mu, label_sigma):
        self.C = classification_model
        self.R = regression_model
        self.feature_mu = feature_mu
        self.feature_sigma = feature_sigma
        self.label_mu = label_mu
        self.label_sigma = label_sigma
        self.R.eval()

    def predict(self, X):
        # 1. 使用分類模型 C(x) 進行預測
        class_preds = self.C.predict(X)
        # 2. 使用迴歸模型 R(x) 進行預測
        X_tensor = torch.tensor(X, dtype=torch.float32)
        X_scaled = (X_tensor - self.feature_mu) / self.feature_sigma # 標準化輸入
        with torch.no_grad():
            temp_preds_scaled = self.R(X_scaled)

        # 反標準化，還原成原始溫度尺度
        temp_preds = (temp_preds_scaled * self.label_sigma + self.label_mu).numpy().flatten()
        # 3. 根據分類結果組合輸出
        final_preds = np.full(X.shape[0], -999.0)
        valid_indices = (class_preds == 1)
        final_preds[valid_indices] = temp_preds[valid_indices]
        return final_preds
```

B. Apply your model to the dataset and verify that the piecewise definition works as expected.

當 C(x) 預測為「無效」時，h(x) 的輸出確實是 -999。

當 C(x) 預測為「有效」時，h(x) 的輸出是一個正常的溫度值。

	經度 (Lon)	緯度 (Lat)	C(x) 預測 (區域)	h(x) 最終預測
0	120.5	23.5	無效	-999.0
1	121.0	24.0	無效	-999.0
2	118.5	21.0	無效	-999.0
3	122.0	25.5	無效	-999.0

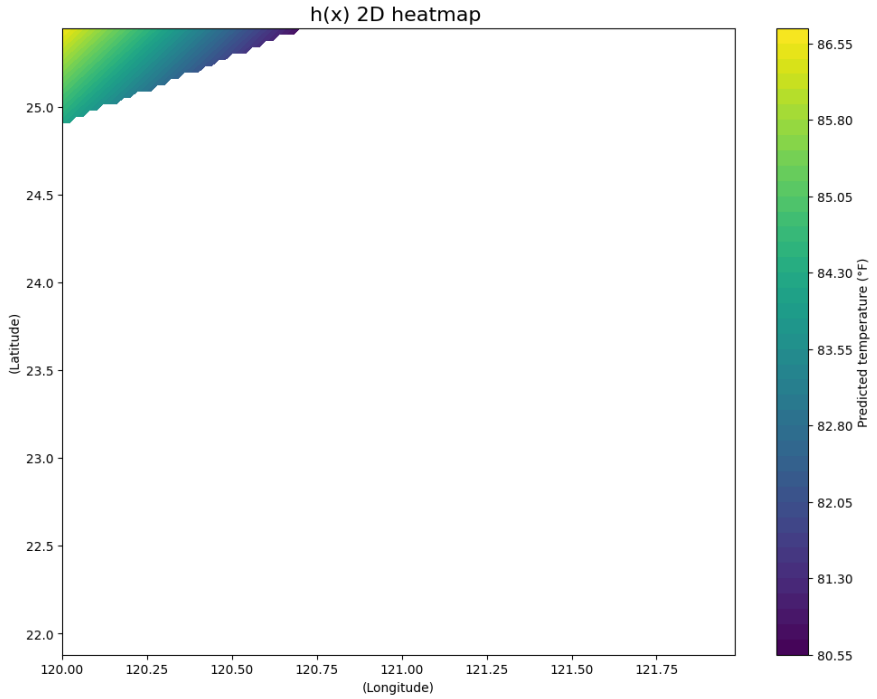
C. Briefly explain how you built the combined function.

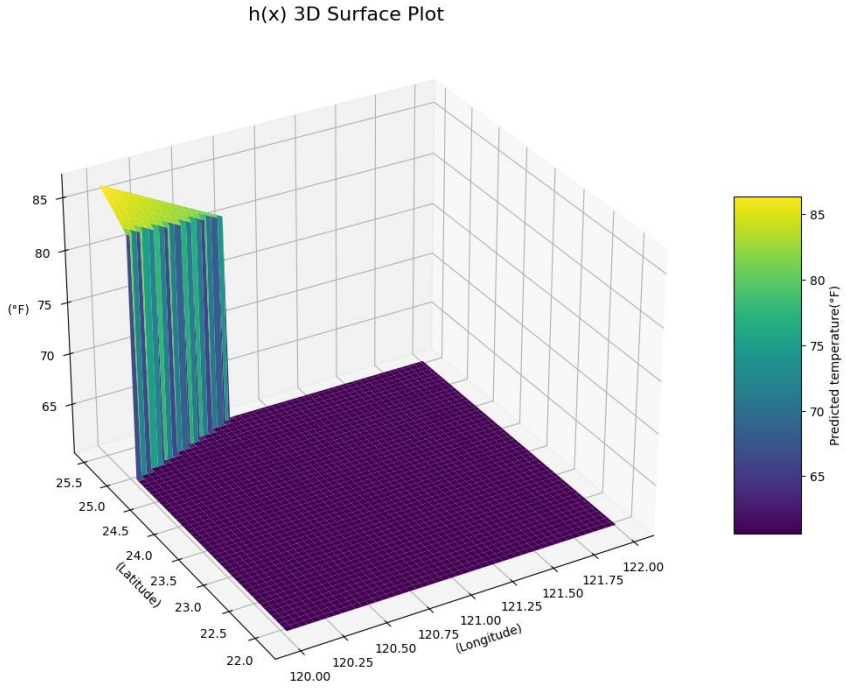
將 GDA 模型作為分類器 C(x)。

將 PyTorch 線性模型作為迴歸器 R(x)。

建立一個 h(x)，它接收一個座標，先交給 C(x) 判斷，再根據判斷結果決定是回傳 -999 還是 R(x) 的預測值。

D. Include plots or tables that demonstrate the behavior of your model.





3. 結論

準確率很低，基本接近用猜的。

原因可能出在 GDA 假設每個類別的資料都服從多變數常態分佈。

檢查原始資料後發現是一個 120*67 的 GRID，

label 1：構成位於中央的實心矩形

label 0：構成了一個圍繞著中央矩形的中空框架

資料並沒有符合假設，然後μ_1和μ_0幾乎重疊，導致模型無法判斷是和哪個近，因此才會有這樣的訓練結果。