Programming assignment

1. Classification

A. Write your own code to implement the GDA algorithm

```python
class QDA_from_scratch:
    def __init__(self):
        self.priors = {}
        self.means = {}
        self.covs = {}
        self.classes = None

    def fit(self, X, y):
        self.classes = np.unique(y)
        n_samples = X.shape[0]
        for c in self.classes:
            X_c = X[y == c]
            # 1. 計算先驗機率 P(y=c)
            self.priors[c] = X_c.shape[0] / n_samples
            # 2. 計算平均向量 μ_c
            self.means[c] = np.mean(X_c, axis=0)
            # 3. 計算協方差矩陣 Σ_c
            # rowvar=False 表示每行為一個樣本，每列為一個特徵
            self.covs[c] = np.cov(X_c, rowvar=False)

    def predict(self, X):

        n_samples = X.shape[0]
        n_classes = len(self.classes)

        log_discriminants = np.zeros((n_samples, n_classes))

        for c in self.classes:
            prior = self.priors[c]
            mean = self.means[c]
            cov = self.covs[c]

            # 計算協方差矩陣的行列式和逆矩陣
            sign, logdet = np.linalg.slogdet(cov)
            inv_cov = np.linalg.inv(cov)

            # 計算每個樣本的對數判別分數
            for i, x_i in enumerate(X):
                diff = (x_i - mean).reshape(-1, 1)
                term1 = -0.5 * logdet
                term2 = -0.5 * (diff.T @ inv_cov @ diff)
                term3 = np.log(prior)
                log_discriminants[i, int(c)] = term1 + term2 + term3
        return self.classes[np.argmax(log_discriminants, axis=1)]
```

依序計算$label_1$和$label_0$的先驗機率、平均向量和 Covariance Matrix，接著利用貝氏定理計算後驗機率來預測，為了穩定有多取了 log 。

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 Covariance Matrix ），結合先驗機率 p(y)，最終使用貝氏定理來預測新資料點。

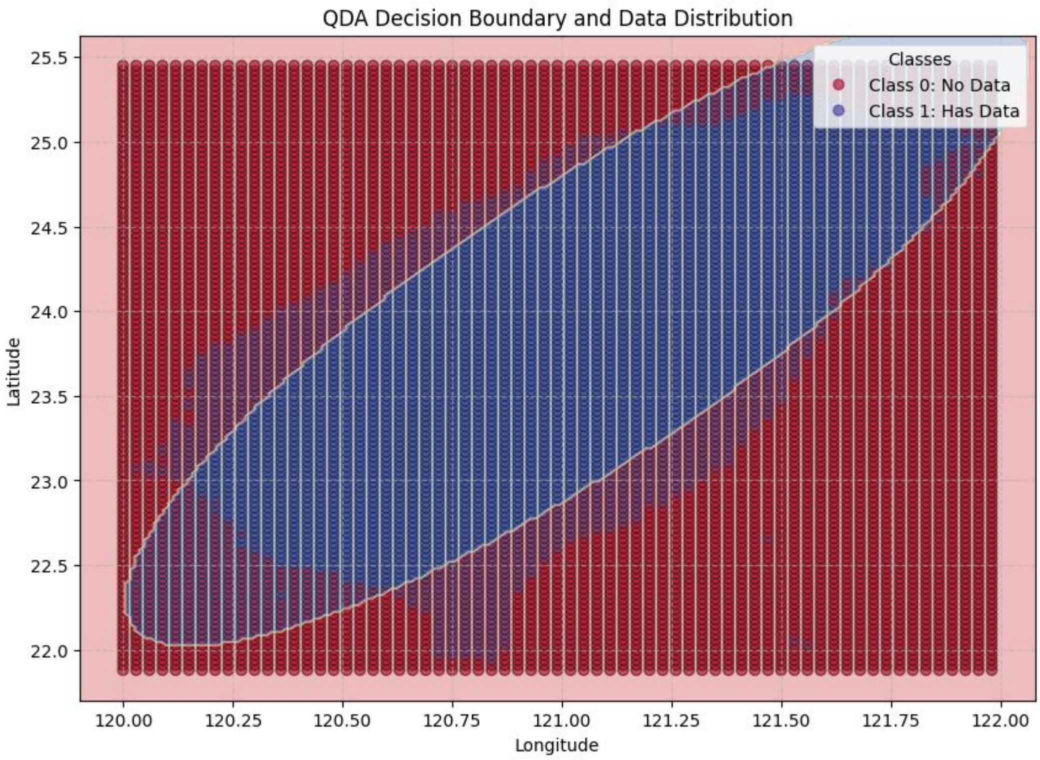
因為資料室地理座標(群聚)，因此可以使用 GDA。

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

將資料切成 8:2 的訓練集和測試集並且使用 Stratified Sampling 保障類別比例和原始數據集相同。

使用 Accuracy 作為評斷指標，GDA 模型在測試集上的準確率: 82.21%

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



2. Regression

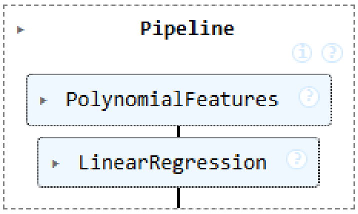
A. Implement this combined model in code

```python
C_x = QDA_from_scratch()
C_x.fit(X_all, y_class)
```

R(x)

```python
is_valid_temp = (temps_all != -999.0)
X_reg = X_all[is_valid_temp]
y_reg = temps_all[is_valid_temp]

degree = 4
R_x = make_pipeline(PolynomialFeatures(degree), LinearRegression())
R_x.fit(X_reg, y_reg)
```



組合模型 h(x)

```python
def h(X, classifier, regressor):
        class_predictions = classifier.predict(X)
        regression_predictions = regressor.predict(X)
        final_output = np.full(X.shape[0], -999.0)
        is_class_1 = (class_predictions == 1)
        final_output[is_class_1] = regression_predictions[is_class_1]
        return final_output
```

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

從真實資料中選取了兩個點進行測試

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。

```
測試點 (內 - 中心點) [120.97071245  23.74593133]: h(x) 輸出 = 13.84
測試點 (外) [120.      21.88]: h(x) 輸出 = -999.00
```

C. Briefly explain how you built the combined function.

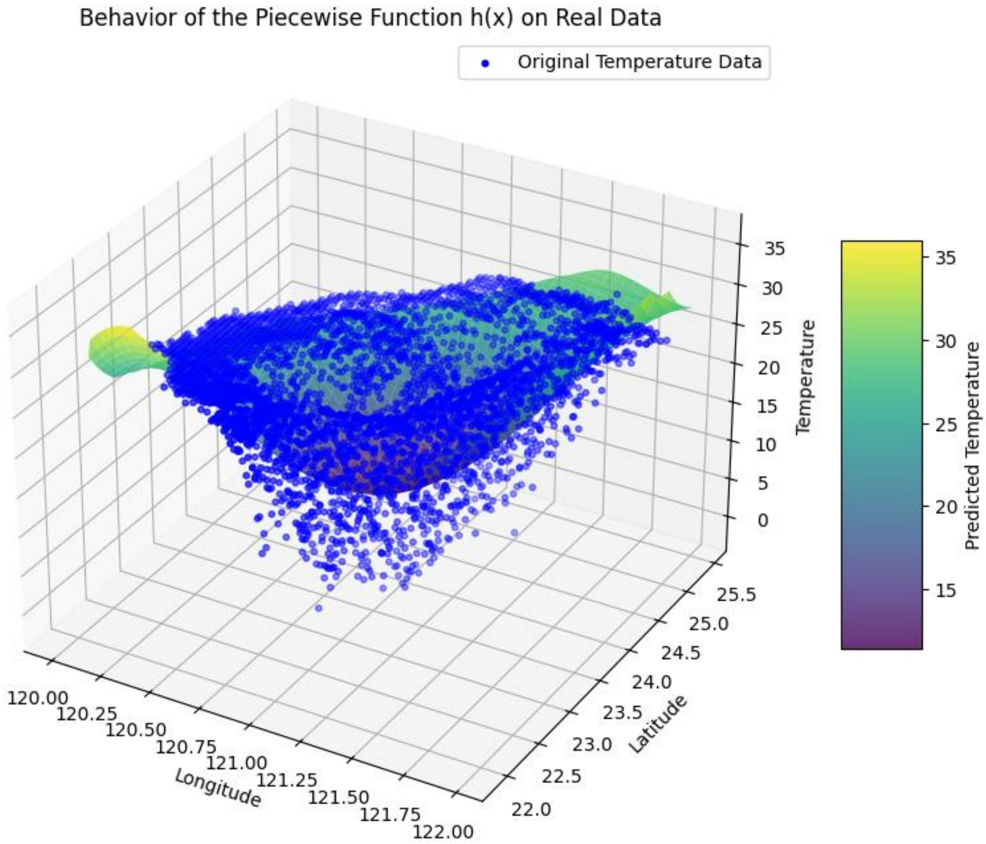
從第一題訓練的模型，C(x)就可分辨有效溫度和無數據，用來定義 h(x)的邊界。

接著選擇使用 4th-degree Polynomial Regression 作為 R(x)。

對任何輸入座標 ，首先呼叫 C(x) 進行分類。 若 C(x) 預測結果為 1，則呼叫 R(x) 預測溫度，並將此溫度作為輸出。 若 C(x) 預測結

果為 0，則直接輸出 -999。

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



藍色點：代表從 XML 檔案中讀取的、所有溫度不為 -999 的真實資料點。

彩色曲面 (Predicted Temperature Surface)：R(x) 學到的平滑溫度預測曲面。

分段邊界 (Piecewise Boundary)：曲面的清晰邊界是由 QDA 分類器 C(x)學習的。在邊界內部，h(x) = R(x)，

平滑的溫度曲面；在邊界外部，h(x) = -999，因此沒有任何曲面被繪製出來。