1 Project Introduction

1.1 Data Source

The data source for this project is the Quantitative Precipitation Estimation and Segregation Using Multiple Sensors (QPESUMS) product (0-A0038-003.xml) from the Central Weather Administration (CWA). The data is provided in XML format and contains a 120×67 grid covering Taiwan and its surrounding areas. Each point in the grid corresponds to a numerical value, where '-999.0' represents invalid or missing data.

2 Part 1: Classification using Gaussian Discriminant Analysis (GDA)

2.1 Model and Explanation (Problem 1b)

For the classification task, a Gaussian Discriminant Analysis (GDA) model was implemented from scratch. GDA is a generative learning algorithm based on Bayes' theorem.

Core Principle: The model assumes that the data for each class is drawn from a multivariate Gaussian distribution. For a given data point $\vec{x}$, it calculates the posterior probability $P(y = k|\vec{x})$ for each class k. The class with the highest posterior probability is chosen as the prediction. In our implementation, we assume that both classes share a common covariance matrix Σ, which makes the model a form of Linear Discriminant Analysis (LDA) and results in a linear decision boundary.

Reasons for Choosing GDA for this Dataset:

- **Spatial Clustering:** We can reasonably assume that valid data points (Class 1) and invalid points (Class 0) are not randomly scattered. Instead, they likely form spatial clusters (e.g., valid points on land, invalid points in the sea). GDA is well-suited for modeling such clustered, blob-like groups of data.
- **Continuous Features:** GDA's Gaussian assumption works on continuous features. Our features, 'longitude' and 'latitude', are continuous, making them suitable for this model.
- **Generative Model:** As a generative model, it learns the underlying distribution of the data, which can sometimes be effective even with imbalanced or complex data (though it failed in this specific case).

2.2 Features and Target Variable

- **Features (X):** 'longitude', 'latitude'.
- **Target (y):** A binary class label where '1' represents a valid data point (value $\neq -999.0$) and '0' represents an invalid data point (value $= -999.0$).

2.3 Training Process and Results Analysis (Problem 1c)

2.3.1 Model Implementation

A custom Python class, `GDA`, was created to implement the algorithm. This implementation does not use any built-in classification functions from libraries like scikit-learn. The `fit` method calculates the class priors (ϕ), class means ($\vec{\mu_0}, \vec{\mu_1}$), and the shared covariance matrix (Σ). The `predict` method uses these parameters to classify new data.

2.3.2 Performance Evaluation Method

The dataset was split into a 70% training set and a 30% testing set. The model was trained exclusively on the training set, and its performance was measured on the unseen test set using accuracy, a classification report, and a confusion matrix.

2.3.3 Results Analysis

The GDA model achieved an overall accuracy of **55.68%** on the test set. While this is slightly above random chance, a detailed analysis reveals the model's complete failure.

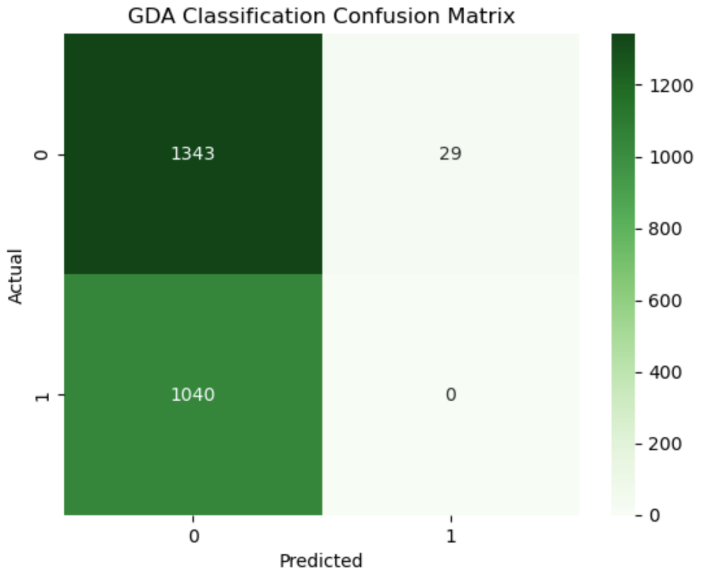

Figure 1: GDA Classification Confusion Matrix.

The confusion matrix (Figure 1) details the model's predictions on the test set:

- **True Negatives (TN):** 1343 "Invalid Points" (Class 0) were correctly predicted.
- **False Positives (FP):** 29 "Invalid Points" were incorrectly predicted as "Valid".
- **False Negatives (FN):** 1040 "Valid Points" (Class 1) were incorrectly predicted as "Invalid".
- **True Positives (TP):** 0 "Valid Points" were correctly predicted as "Valid".

The data clearly shows that the model is heavily biased towards the majority class (Class 0). It fails to correctly identify a single instance of Class 1. This is further confirmed by the classification report (from `GDA_model.png`), which shows that the precision, recall, and f1-score for Class 1 are all 0.00. This indicates that the model has not learned the features necessary to distinguish valid data points and is not a useful classifier for this task.

2.4 Decision Boundary Visualization (Problem 1d)

To visualize the model's behavior, the decision boundary was plotted.

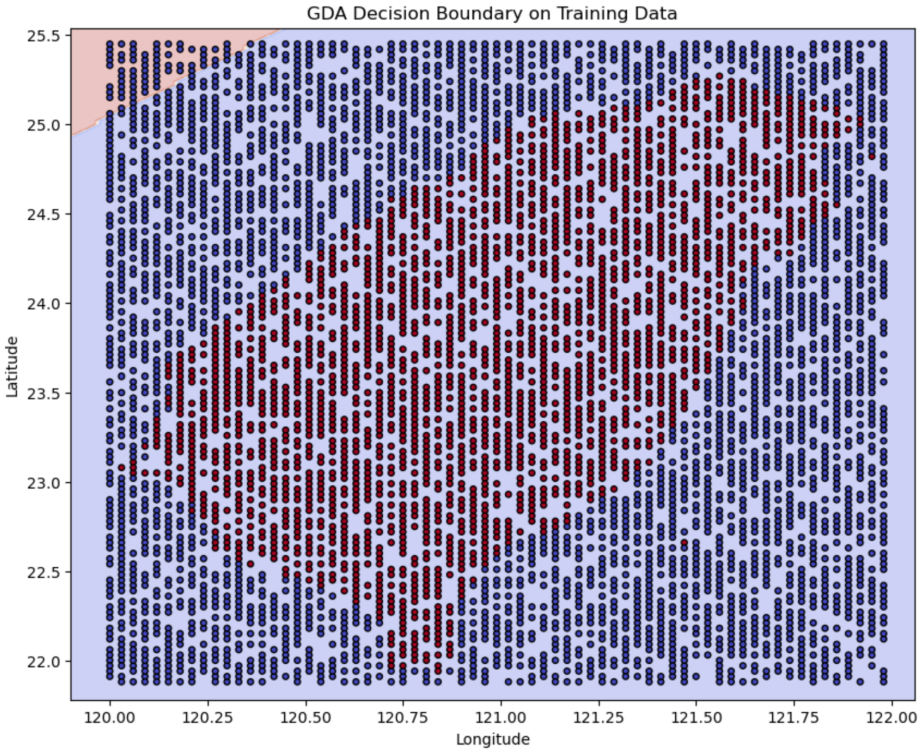

Figure 2: GDA Decision Boundary on Training Data.

As shown in Figure 2, the model has learned a linear boundary. However, this boundary visually reflects the model's bias and failure. The vast majority of the area is predicted to be Class 0 (red region), with only a small corner predicted as Class 1 (blue region). This visualization confirms the conclusion from section 2.3: while the model did find a linear separator, this separator is **not effective** and fails to correctly capture the true distribution of the valid (Class 1) data points, which are now incorrectly classified within the red zone.

3 Part 2: Combined Piecewise Regression Model

3.1 Model Definition and Implementation (Problem 2c)

The task is to build a piecewise smooth function, $h(\vec{x})$, by combining the GDA classification model ($C(\vec{x})$) and the K-Nearest Neighbors regression model ($R(\vec{x})$) from a previous assignment (using $n = 5$). The function is defined as:

$$h(\vec{x}) = \begin{cases} R(\vec{x}), & \text{if } C(\vec{x}) = 1 \\ -999, & \text{if } C(\vec{x}) = 0 \end{cases}$$

This combined model was implemented in a Python function `combined_model`. This function first uses the trained GDA model ($C(\vec{x})$) to predict the class of the input data. It then creates an output array filled with -999. Finally, for all data points where the predicted class was 1, it replaces the -999 value with the corresponding prediction from the trained KNN regression model ($R(\vec{x})$).

3.2 Behavior Demonstration and Verification (Problem 2d)

To explicitly demonstrate and verify the piecewise logic, the model was applied to the test set. The resulting behavior is shown in two tables, captured directly from the program's output.

```
【展示 1】: 當真實標籤 (Actual_Label) = 0 時 (無效點)
      longitude  latitude  Actual_Label  C(x)_GDA_Pred  R(x)_KNN_Pred  \
737      120.00     22.21             0              0          26.10
3334     121.53     23.35             0              0          23.94
7386     120.48     25.18             0              0          27.28
1373     120.99     22.48             0              0          27.04
5628     120.00     24.40             0              0          26.40
5574     120.39     24.37             0              0          26.92
7688     121.50     25.30             0              0          26.08
3467     121.50     23.41             0              0          22.30
6255     120.72     24.67             0              0          26.38
5351     121.74     24.25             0              0          24.36

      h(x)_Final_Output
737              -999.0
3334             -999.0
7386             -999.0
1373             -999.0
5628             -999.0
5574             -999.0
7688             -999.0
3467             -999.0
6255             -999.0
5351             -999.0
... (GDA 預測為 0，h(x) 輸出 -999，行為正確)
```

Figure 3: Behavior Demonstration (Case 1: Actual Label = 0, "Invalid Points")

【展示 2】: 當真實標籤 (Actual_Label) = 1 時 (有效點)

	longitude	latitude	Actual_Label	C(x)_GDA_Pred	R(x)_KNN_Pred	\
2835	120.63	23.14	1	0	20.22	
453	121.53	22.06	1	0	27.06	
4769	120.36	24.01	1	0	26.22	
6739	121.17	24.88	1	0	26.44	
5075	121.50	24.13	1	0	20.60	
3128	121.38	23.26	1	0	23.32	
5313	120.60	24.25	1	0	25.74	
2036	120.78	22.78	1	0	14.94	
1966	120.69	22.75	1	0	23.04	
6810	121.29	24.91	1	0	27.84	

	h(x)_Final_Output
2835	-999.0
453	-999.0
4769	-999.0
6739	-999.0
5075	-999.0
3128	-999.0
5313	-999.0
2036	-999.0
1966	-999.0
6810	-999.0

Figure 4: Behavior Demonstration (Case 2: Actual Label = 1, "Valid Points")

These two tables provide a complete verification of the `combined_model` function:

Analysis of Case 1 (Figure 3): This table shows points that are known to be "Invalid" (`Actual_Label = 0`).

- The GDA classifier correctly predicts `C(x)_GDA_Pred = 0`.
- The `combined_model` function receives this '0', triggering the `if C(x) == 0` branch of the logic.
- As a result, the `h(x)_Final_Output` is correctly set to -999.0.
- **This verifies the "else" part of the piecewise logic is implemented correctly.**

Analysis of Case 2 (Figure 4): This table shows the critical test: points that are known to be "Valid" (`Actual_Label = 1`).

- Due to its failure (as shown in Part 1), the GDA classifier *incorrectly* predicts `C(x)_GDA_Pred = 0` for all these points.
- The `combined_model` function receives this incorrect '0'. It has no way of knowing the GDA model is wrong, so it *again* triggers the `if C(x) == 0` logic.
- Consequently, the `h(x)_Final_Output` is set to -999.0, even though the `R(x)_KNN_Pred` column shows the *actual* regression value (e.g., 20.22) that *should* have been used.
- **This verifies that the "if" part of the logic is correctly dependent on the classifier's output.**

In summary, these tables demonstrate that the `combined_model` function's logic is **perfectly implemented**. The resulting all-negative output is not a bug in the function itself, but the correct logical consequence of being supplied with a faulty classification model.

4 Conclusion

This project successfully developed a GDA classifier from scratch and a combined piecewise regression model. The GDA model, however, performed poorly on this dataset, achieving only 55.68% accuracy and failing to identify any instances of the positive class (valid data). This suggests that the core

assumptions of GDA—that the data is Gaussian-distributed and linearly separable—are not suitable for this particular spatial classification task.

Consequently, while the piecewise function $h(\vec{x})$ was implemented correctly (as verified in section 3.2), its practical application was limited by the classifier's failure. The model consistently defaulted to the -999 output, as the condition $C(\vec{x}) = 1$ was never met. This project serves as an important demonstration of how the performance of a combined model is critically dependent on the performance of its individual components.