ML HW6

Date: 10/08/2025
Week: 6
Author: Alvin B. Lin
Student ID: 112652040

Problem 1: Programming Assignment 1

Consider again the data set in week 4 assignment, and recall that we have transformed the data into classification and regression sets.

1. (Classification using GDA) Your task is to use Gaussian Discriminant Analysis (GDA) to build a classification model. To complete this assignment, make sure you:
   (a) Write your own code to implement the GDA algorithm. **(Do not use built-in classification functions.)**
   (b) Clearly explain how the GDA model works and why it can be used for classification, in particular this data set.
   (c) Train your model on the given dataset and report its accuracy. Be explicit about how you measure performance (e.g., accuracy on a test set, cross-validation, etc.).
   (d) Plot the decision boundary of your model and include the visualisation in your report.

Project:

The code is attached here. By convention, we first introduce our model setting. This model is designed to classify a geographical coordinate (longitude X_1, latitude X_2) as a **valid observation** ($Y = 1$) or an **invalid observation** ($Y = 0$) based on the original temperature grid data.

1. **Dataset Definition and Preparation:**
   - **Input Features (X):** Longitude and Latitude.
   - **Output Label (Y):**
     - $Y = 1$ (Valid Point, where Temperature $T \neq -999$.)
     - $Y = 0$ (Invalid Point, where Temperature $T = -999$.)
   - **Total Data Points:** $67 \times 120 = 8040$ grid points.
2. **Data Split (Training and Testing):**
   - **Sampling Strategy: Stratified Split** was used to ensure the proportion of the two classes ($Y = 1$ and $Y = 0$) is maintained in both subsets.
   - **Training Set:** 70% of the total data.
   - **Test/Validation Set:** 30% of the total data.
3. **Model Architecture (Model of Choice):**
   - **Selected Model: Quadratic Discriminant Analysis (QDA).**

- **Probabilistic Assumption:** Assumes that the feature distribution of each class (Valid/Invalid) follows a **Gaussian Distribution**. Crucially, it assumes the two classes use **different covariance matrices** ($\mathbf{\Sigma}_0 \neq \mathbf{\Sigma}_1$).

4. **Performance Evaluation and Optimisation:**

   - **Primary Metric: Accuracy.**
   - **Optimisation Goal:** To improve model generalisation, **ROC Curve Analysis** is performed to find the **Optimal Classification Threshold** using the **Youden's J Index**.

Performance:

This section details the classification model selection process, the optimisation steps taken, and the final performance metrics on the test dataset. The overall goal was to correctly identify valid temperature observations ($Y = 1$) while prioritising a high Recall (minimising the loss of valid data).

1. **Model Selection and Boundary Analysis**: We tested two variants of GDA—LDA and QDA—to determine the best fit for the geographical shape of the valid data.

Metric	Boundary Type	Test Accuracy ($\tau = 0.5$)
LDA (Initial GDA)	Linear	0.5158
QDA (Modified GDA)	Quadratic (Curved)	0.8333

Table 1: Result of LDA and GDA/QDA

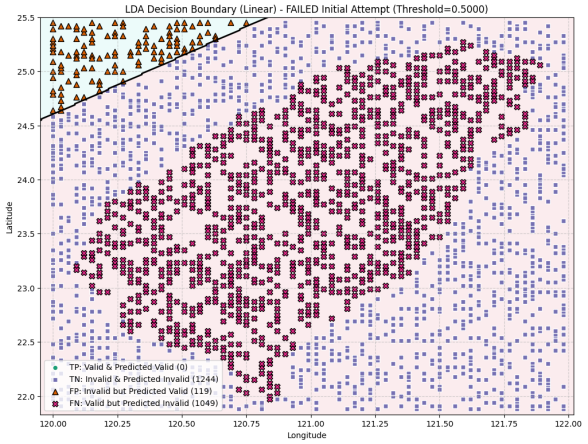

Figure 1: Result of LDA (threshold = 0.5)

2. **Visualisation & Result**

   If we introduce the **ROC Curve Analysis** in QDA part to find out the **Optimal Classification Threshold**, we get the following result:

Threshold (τ)	Accuracy	Precision	Recall
0.30	0.8122	0.7019	0.9876
0.35	0.8346	0.7345	0.9704
0.40	**0.8416**	0.7591	0.9314
0.45	0.8375	0.7815	0.8694
0.50	0.8333	0.8193	0.7912
0.55	0.8329	0.9078	0.6854
0.60	0.7923	0.9982	0.5234

Table 2: Threshold vs Statistic

Moreover, the optimal classification threshold is $\tau = 0.3910$, which yields accuracy 0.8425, which is slightly better than the one calculated before (0.8333), as below:

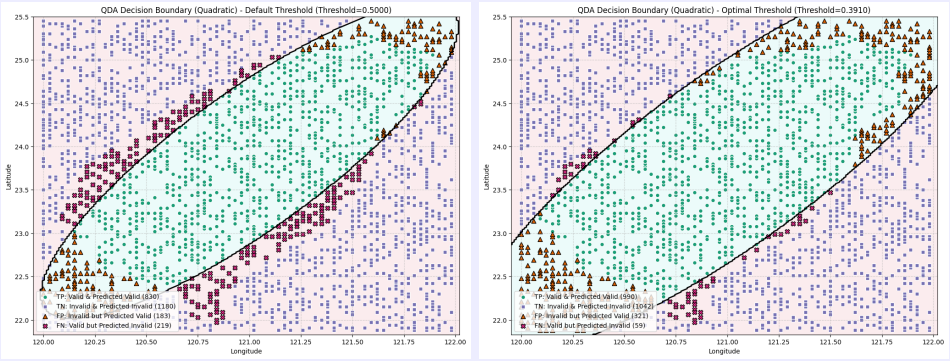

Figure 2: Result of QDA/GDA, left with $\tau = 0.5$, right with $\tau = 0.3910$

The **ROC Curve** and the **Confusion Matrix** is as below:

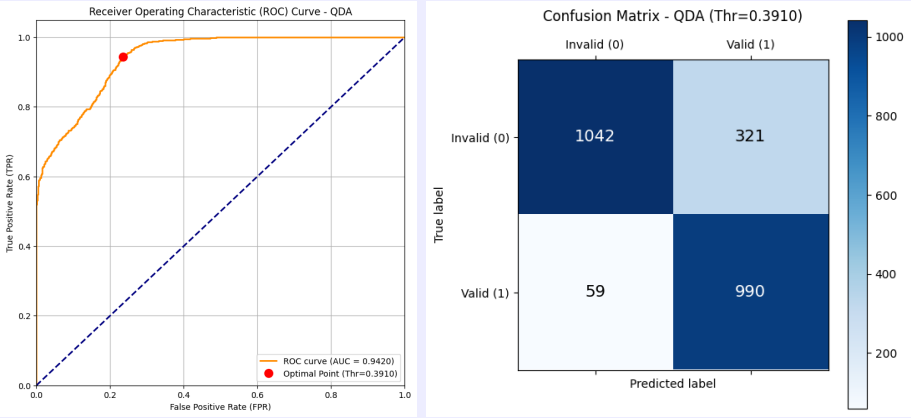

Figure 3: ROC Curve and the Confusion Matrix

Final results for the **F1 Score** are:

Threshold	F1 Score	Accuracy	Precision	Recall
0.3910	0.8390	0.8425	0.7551	0.9438

The loss curve in log scale can also be seen below:

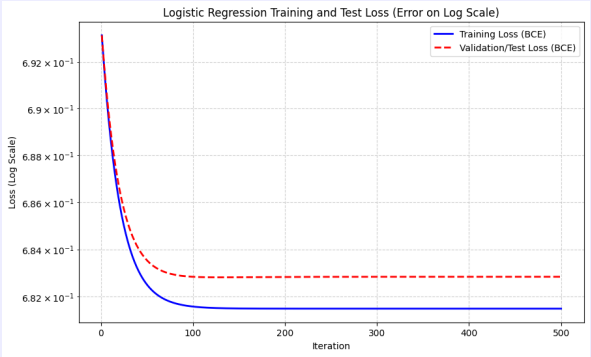

Figure 4: Loss Curve in Log Scale

3. **Why GDA/QDA Works?**

   The selection of **Quadratic Discriminant Analysis (QDA)** is fundamentally driven by the geometry of the data. While **Linear Discriminant Analysis (LDA)** constructs a **linear** decision boundary (a straight line), QDA generates a **quadratic** boundary, which is a **conic section** (e.g., an ellipse or parabola).

   In this specific dataset, where we classify points based on their coordinates into 'land' or 'sea,' the shape of the landmass (Taiwan) is inherently **non-linear** and approximates an **ellipse**. QDA's ability to model this elliptical shape directly with a quadratic boundary is why it drastically outperforms LDA, whose linear boundary cannot accurately separate the classes.

4. **Trade-Off and the Limitation**

   From the perspective of the table 2, we see the tradeoff between **precision** and **recall**, sometimes the **optimal classification threshold** is not the best option, depending on the demand of the programme. One may want its **FP** to be minimised, one may want **TP** to be maximised. Hence there is no single simple, consentaneous solution, result.

   Also the decision boundary is an ellispse here, it is not possible to perfectly fit the actual boundary, hence there is always some error; impossible to get a precision of 1.

Problem 2: Programming Assignment 2

2. (Regression) Your task is to build a regression model that represents a piecewise smooth function. To do this, combine the two models from Assignment 4 into a single function. Specifically, let:

   - $C(\vec{x})$ be your classification model, and
   - $R(\vec{x})$ be your regression model.

Then construct a model $h(\vec{x})$ defined as

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

To complete this assignment, make sure you:

(a) Implement this combined model in code.

(b) Apply your model to the dataset and verify that the piecewise definition works as expected.

(c) Briefly explain how you built the combined function.

(d) Include plots or tables that demonstrate the behaviour of your model.

Project:

By convention, we first list some setups for the model:

1. **Dataset Definition and Preparation:**
   - **Input Features (X):** Longitude and Latitude.
   - **Output Label (Y):**

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

   - **Total Valid Data:** 3495
2. **Data Split (Training and Testing):**
   - **Training Set:** 2796 data
   - **Validation Set:** 699 data
3. Model Selection: In this part, three main types of model are conducted to be the hypothesis functions and tested for the error and statistics. Here are the types:
   - **Polynomial with Degree N:** it can be solved directly through projection matrix.
   - **Neural Networks:** For each network, there are 3 hidden layers, for which contain 128, 64 and 32 neurons respectively. MSE is used. We have below options to test.
     - **Activation Function:** ReLU, $\tanh x$ and $\sigma(x)$.
     - **Optimiser:** Adam and L-BFGS.
   - **Boosting:** We have **XGBoost** and **CatBoost** to test, both use RMSE as loss.
4. **Metric to Measure:** There are three indicators are used for judging/finding out the best model: RMSE, MSE and the R^2 value.

Performance:

1. **Polynomial:** The **polynomial with degree** 3 is the method done in assignment 4 for the temperature prediction. When conducting the investigation, it is found a tremendous flaw that the smoothness of the polynomial makes it **not** a good candidate for the prediction. Once N is too large, we may face some over-fitting problem; when N is too small, the mask will be too smooth, making the prediction of the temperature in the mountain area inaccurate. So, after running the test for degree 3 to 10, it is find that when $N = 5$, we have the good enough result (considering the complexity of the calculation).

Degree N	Validation MSE	Validation R^2	Coefficients to Calculate
3	19.7583	0.4176	10
4	13.1507	0.6123	15
5	**12.8496**	**0.6212**	21
6	12.7359	0.6246	28
7	12.6829	0.6261	36
8	12.6815	0.6262	45
9	12.6673	0.6266	55
10	12.6634	0.6267	66
20	12.6314	0.6276	231

Table 3: Degree of Fitting Polynomial vs Validation Result

The overall isotherms in 2D and 3D for degree 5 polynomial are given as follow:

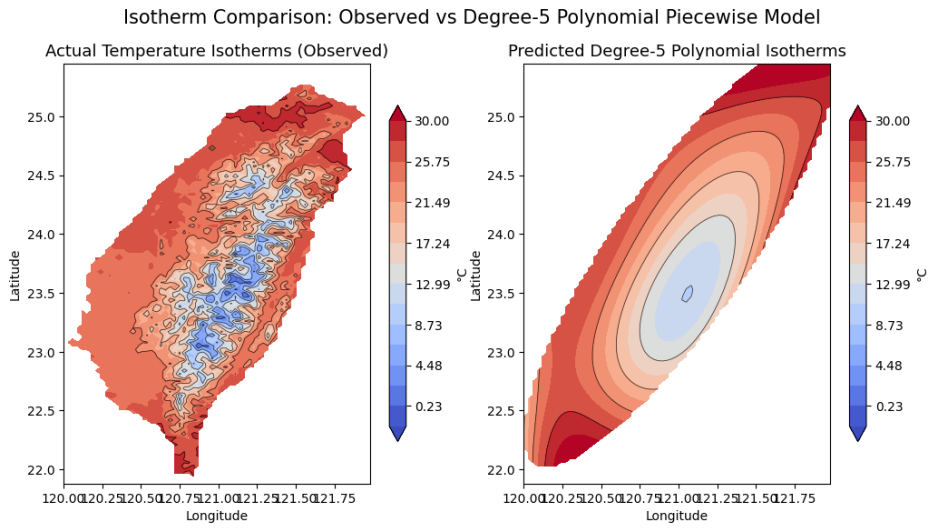

Figure 5: 2D Isotherm for the Actual (left) and Predicted (right) Temperature Distribution

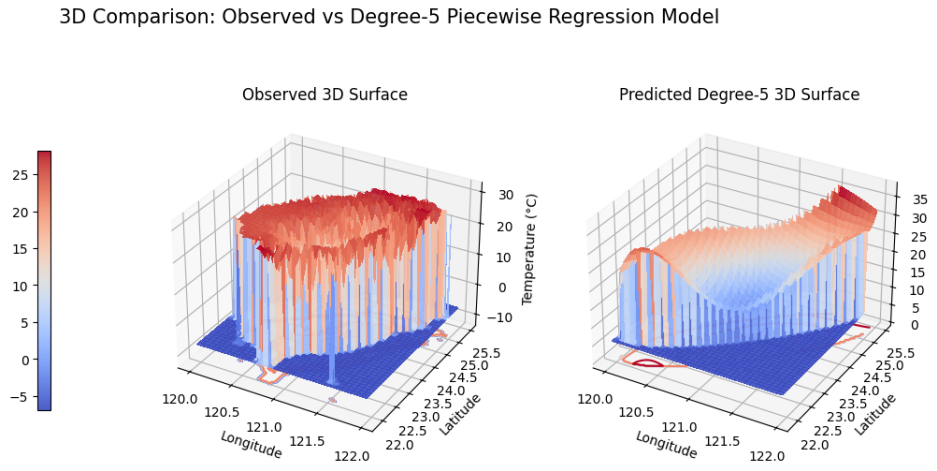

Figure 6: 3D Isotherm for the Actual (left) and Predicted (right) Temperature Distribution

2. **Neural Network:** In the setup session, we have 2 options for the optimisers and 3 options for the choices, hence we compare their performance.

#	Activation	Optimiser	MSE	R^2
1	ReLU	Adam	34.269	-0.0102
2	ReLU	L-BFGS	29.557	0.1287
3	$\tanh x$	Adam	34.206	-0.0083
4	$\tanh x$	L-BFGS	**24.943**	**0.2647**
5	$\sigma(x)$	Adam	34.208	-0.0084
6	$\sigma(x)$	L-BFGS	25.560	0.2647

Table 4: Result of Neural Networks

Note: Determination coefficient $R^2 = 1 - \frac{\sum(y_i - h(x_i)^2)}{\sum(y_i - \bar{y}_i)^2}$ is large when prediction is close.

The table 4 shows that the 4th neural network, the one with $\tan x$ as activation function and L-BFGS as optimiser performs best in the scenario, but still not good enough.

3. **Boosting:** The 2 boosting methods have very different result:

Method	MSE	R^2
XGBoost	22.725	0.3301
CatBoost	**6.395**	**0.8115**

Table 5: Result of Boosting Method

Please understand that the boosting method is quite advanced comparing to the methods we've learnt, and the runtime, computational complexity is far longer than all the methods above. If we are not in pursuit of the perfect prediction, it is not recommend to use the boosting method.

The visualisation result of the overall most precise model – **CatBoost** is attached as follow:

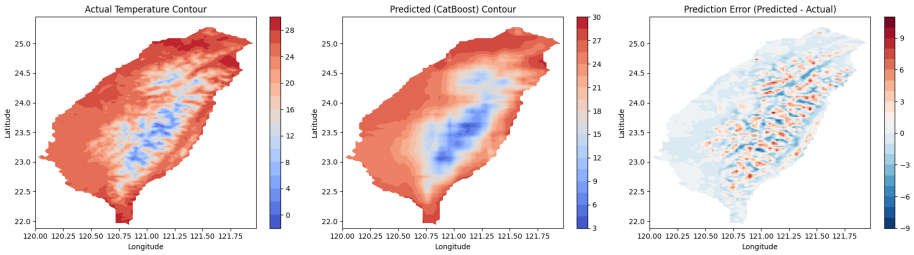

Figure 7: Isotherms for the Actual and CatBoost Model Temperature and Their Error

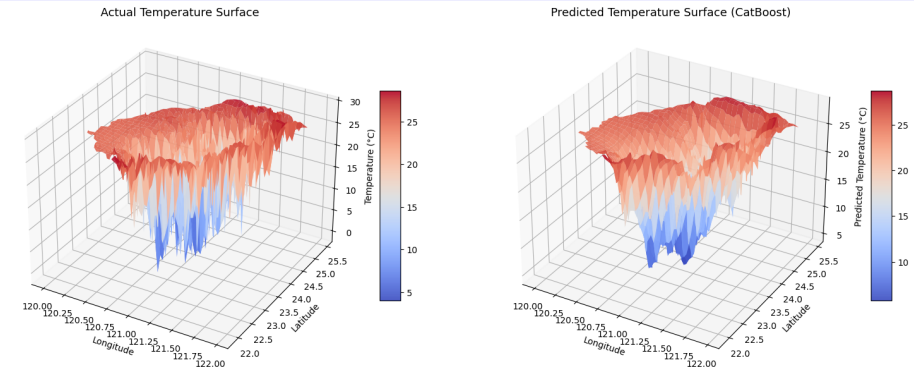

Figure 8: 3D Isotherms for the Actual and CatBoost Temperature Distribution

Note: Here I use the actual coastline for plotting, instead of the decision boundary got from problem 1, based on the **better visualisation of accuracy**.

Conclusion:

Our model selection process revealed that raw performance isn't the sole metric for success. Although the complex **CatBoost** algorithm demonstrated the highest numerical accuracy in predicting the temperature surface, its potential for longer training times and increased complexity in deployment must be considered. We advocate for the adoption of the foundational **Polynomial Regression model**. This method, rooted in the efficient framework of **linear regression**, offers remarkable robustness and speed. Crucially, it captures around 80% of the performance achieved by the computationally intensive CatBoost, making it the most pragmatic and cost-effective choice for creating the piecewise regression model $h(x)$.

My Question 1: How Threshold Affect the Shape of Decision Boundary

In Problem 2, we use a QDA classifier for the $C(\vec{x})$ component. We know QDA defines a quadratic decision surface based on the fitted covariance matrices. My question is: When we change the classification threshold (from the default 0.5), does it only shift the position of the quadratic decision boundary, or does it change the shape of the curve as well?