Machine Learning

313704071 陳安定 Assignment 4

1. Data conversion

Original data from: https://opendata.cwa.gov.tw/dataset/observation/O-A0038-003

The original data is organized into two new data sets, namely (longitude, latitude, label) and (longitude, latitude, value), where the label is 0 or 1 and the value is the corresponding temperature value. The following is example.

```
lon,lat,label
120.0000,21.8800,0
120.0300,21.8800,0
120.0600,21.8800,0
120.0900,21.8800,0
120.1200,21.8800,0
120.1500,21.8800,0
120.1800,21.8800,0
```

Label data

```
lon,lat,value
120.8400,21.9400,29.80
120.7200,21.9700,30.70
120.7500,21.9700,30.70
120.7800,21.9700,30.00
120.8100,21.9700,30.20
120.8400,21.9700,29.10
120.7200,22.0000,30.80
```

Value data

2. Classification model – Logistic Regression

Hypothesis function:

$$h_\theta(x) = \hat{p} = P(y = 1|x)$$

$$h_\theta(x) = \sigma(\theta^T \tilde{x}) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$$y \in \{0,1\}$$

Convert the input features into the probability of belonging to a certain category, and use the sigmoid function to compress the output to between 0 and 1.

$$\hat{p} = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2)}}$$

$\hat{p}$: The probability of predicted $y = 1$

θ_0, θ_1, θ_2: Parameters that the model to learn

x_1, x_2: Input features (x_1: longitude, x_2: latitude)

Threshold:

if $p \geq 0.5$, $\hat{y} = 1$;

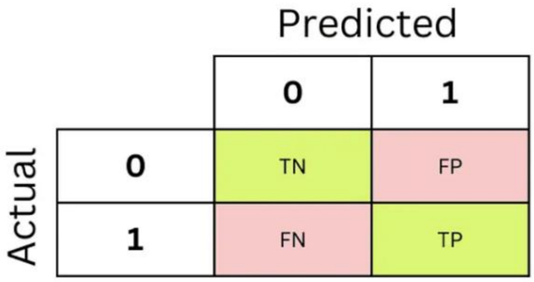
if $p < 0.5$, $\hat{y} = 0$

(1) Directly use the suite in Scikit-learn

```
Accuracy: 0.5733830845771144
Confusion matrix:
 [[922   7]
 [679   0]]
Report:
              precision    recall  f1-score   support

           0       0.58      0.99      0.73       929
           1       0.00      0.00      0.00       679

    accuracy                           0.57      1608
   macro avg       0.29      0.50      0.36      1608
weighted avg       0.33      0.57      0.42      1608
```



Logistic Regression defaults to "**maximizing overall accuracy.**" It finds that "always guessing 0" gives it 57% accuracy, so it simply stops guessing 1.

Meteorological data has strong spatial continuity, but it may be difficult to classify "missing vs. valid" based solely on latitude and longitude distribution.
We try to adjust the model in code: **dealing with class imbalance**.

```
model = LogisticRegression(class_weight="balanced")
```

Then,

```
Accuracy: 0.5472636815920398
Confusion matrix:
 [[503 426]
 [302 377]]
Report:
              precision    recall  f1-score   support

           0       0.62      0.54      0.58       929
           1       0.47      0.56      0.51       679

    accuracy                           0.55      1608
   macro avg       0.55      0.55      0.54      1608
weighted avg       0.56      0.55      0.55      1608
```

(2) Custom Loss

Binary Cross-Entropy Loss:

$$Loss = -y \ln h_\theta(x) - (1-y)\ln\big(1-h_\theta(x)\big)$$

$$y \in \{0,1\},\ \ h_\theta(x) = \hat{p}$$

Gradient descent method:

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta Loss$$

$$= \theta_n - \eta \left\{ \sum_i \frac{y_i - h_\theta(x_i)}{h_\theta(x_i)(1-h_\theta(x_i))} \nabla_\theta h_\theta(x_i) \right\}$$

Through cross-entropy loss + gradient descent, θ_1, θ_2, and θ_0 are continuously adjusted, ultimately obtaining an optimal set of parameters.

1st try:

```
[001] Loss=nan | Train Acc=0.481 | Test Acc=0.750 | Confusion Matrix: [[TN=1206, FP=0], [FN=402, TP=0]]
[020] Loss=nan | Train Acc=0.519 | Test Acc=0.750 | Confusion Matrix: [[TN=1206, FP=0], [FN=402, TP=0]]
[040] Loss=nan | Train Acc=0.519 | Test Acc=0.750 | Confusion Matrix: [[TN=1206, FP=0], [FN=402, TP=0]]
```

Problem: $Loss = NaN$

Possible causes:

log(0)、sigmoid overflow、Learning rate too large…

Solution: change to use **log-sum-exp trick**, avoiding computing **log(sigmoid)**

2nd try:

```
[001] Loss=19.3766 | Train Acc=0.481 | Test Acc=0.250 | Confusion Matrix: [[TN=0, FP=1206], [FN=0, TP=402]]
[020] Loss=17.9454 | Train Acc=0.481 | Test Acc=0.250 | Confusion Matrix: [[TN=0, FP=1206], [FN=0, TP=402]]
[040] Loss=16.4388 | Train Acc=0.481 | Test Acc=0.250 | Confusion Matrix: [[TN=0, FP=1206], [FN=0, TP=402]]
[060] Loss=14.9323 | Train Acc=0.481 | Test Acc=0.250 | Confusion Matrix: [[TN=0, FP=1206], [FN=0, TP=402]]
[080] Loss=13.4257 | Train Acc=0.481 | Test Acc=0.250 | Confusion Matrix: [[TN=0, FP=1206], [FN=0, TP=402]]
[100] Loss=11.9191 | Train Acc=0.481 | Test Acc=0.250 | Confusion Matrix: [[TN=0, FP=1206], [FN=0, TP=402]]
[120] Loss=10.4125 | Train Acc=0.481 | Test Acc=0.250 | Confusion Matrix: [[TN=0, FP=1206], [FN=0, TP=402]]
[140] Loss=8.9059 | Train Acc=0.481 | Test Acc=0.250 | Confusion Matrix: [[TN=0, FP=1206], [FN=0, TP=402]]
[160] Loss=7.3994 | Train Acc=0.481 | Test Acc=0.250 | Confusion Matrix: [[TN=0, FP=1206], [FN=0, TP=402]]
[180] Loss=5.8928 | Train Acc=0.481 | Test Acc=0.250 | Confusion Matrix: [[TN=0, FP=1206], [FN=0, TP=402]]
[200] Loss=4.3865 | Train Acc=0.481 | Test Acc=0.250 | Confusion Matrix: [[TN=0, FP=1206], [FN=0, TP=402]]
```

Loss decreases, but accuracy remains unchanged. This means the prediction probability is moving in the right direction, but has not yet crossed the threshold of 0.5, so the binarization result (二值化結果) is the same.

Logistic Regression is too weak.

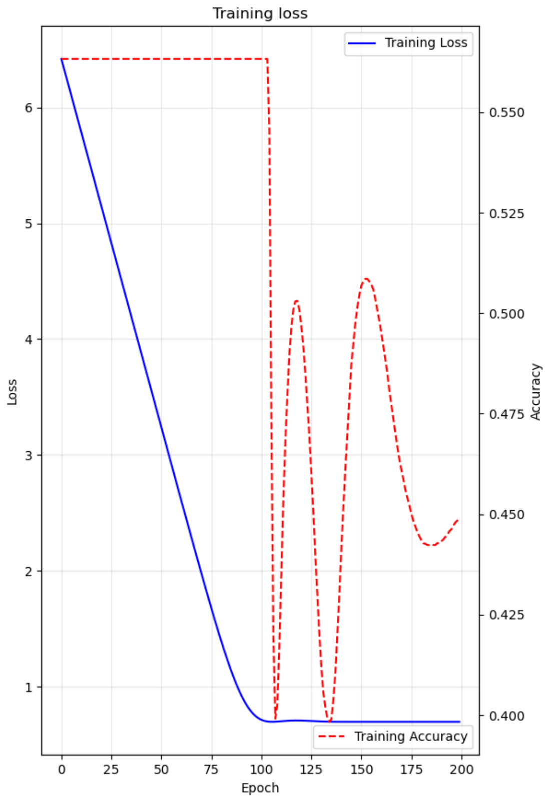
Essentially, it is a "linear decision boundary" (a slash). In this data, the "missing value distribution" on the map may not be cut by a straight line, and there may be large blocks and complex areas.

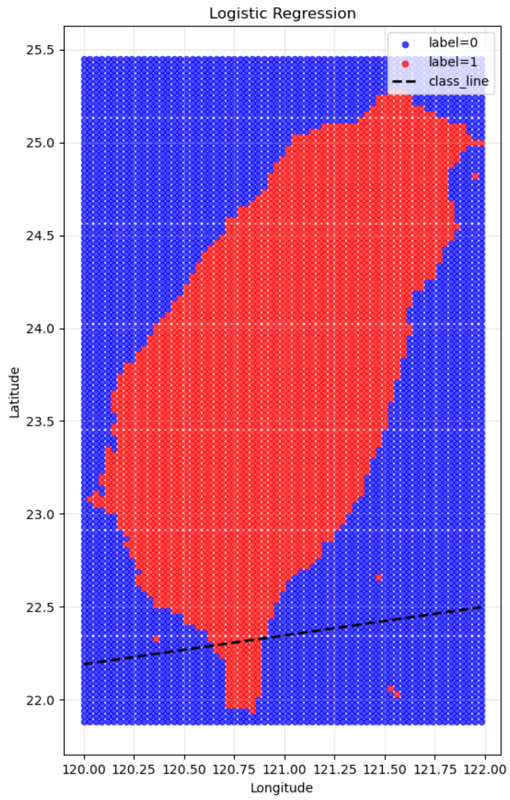
From the previous confusion matrix, we can see that the model achieves around 70% accuracy when choosing to “guess all 0s.” This suggests that 0s are more likely to learn patterns than 1s in the data, while the distribution of 1s may be scattered and irregular.

3rd try:

Inherited and used nn.module of Pytorch, which provides functions such as automatic gradient calculation and parameter management; loss function: BCE.

Function **nn.BCEWithLogitsLoss()** avoids numerical instability (when logits are large or small) More efficient than using sigmoid + BCELoss separately.






```
[001] Loss=6.4185 | Train Acc=0.563
[050] Loss=3.3013 | Train Acc=0.563
[100] Loss=0.7227 | Train Acc=0.563
[150] Loss=0.6965 | Train Acc=0.505
[200] Loss=0.6962 | Train Acc=0.449
```

Although the loss can be reduced to around 0.6, the prediction accuracy has decreased instead of increased. This is because accuracy is a step function that depends on whether the prediction probability crosses the 0.5 threshold. Even a slight parameter update, such as one that causes the prediction probability of certain points to fluctuate between 0.49 and 0.51, will cause the accuracy to fluctuate.

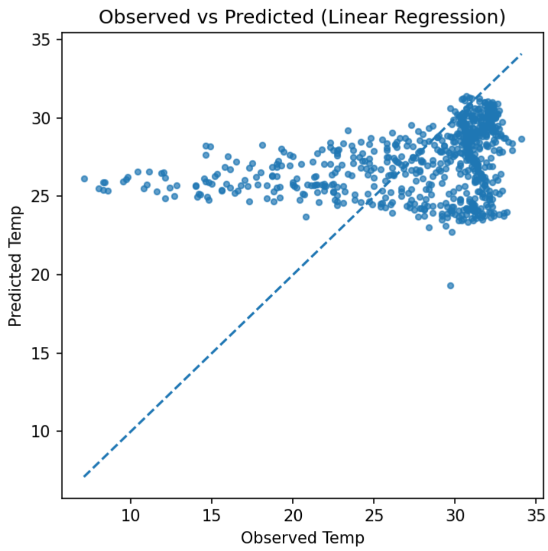
This is not a purely linearly separable data type, so a straight line cannot consistently separate the red and blue areas.

3. Regression model - Linear regression

$$h_\theta(x) = \hat{T} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

x_1: longitude, x_2: latitude

Result:



Bash line: Observed Temp = Predicted Temp **(perfect prediction)**

Most points do not follow the dotted line, and the predicted range is much narrower than the actual temperature (mostly around 24–31°C). When the temperature is low (x is small), the prediction is high; when the temperature is high (x is large), the prediction is low.

Typical "Mean Reversion" behavior of linear, low-feature models.

```
Coefficients of lon & lat : [-5.1195173   2.93689361]
Intercept : 576.9318
MAE : 4.0329
RMSE: 5.2543
R^2  : 0.0843
```

$$\hat{T} = 576.9 - 5.1x_1 + 2.9x_2$$