DM2024 ISA5810 Lab2 Homework – Kaggle Competition

1. 數據處理流程

1.1 資料提取

	tweet_id	hashtags	text	identification
0	0x376b20	[Snapchat]	People who post "add me on #Snapchat" must be ...	train
1	0x2d5350	[freepress, TrumpLegacy, CNN]	@brianklaas As we see, Trump is dangerous to #...	train
3	0x1cd5b0	[]	Now ISSA is stalking Tasha 😂😂😂 <LH>	train
5	0x1d755c	[authentic, LaughOutLoud]	@RISKshow @TheKevinAllison Thx for the BEST TI...	train
6	0x2c91a8	[]	Still waiting on those supplies Liscus. <LH>	train
...	...	...	...	...
1867526	0x321566	[NoWonder, Happy]	I'm SO HAPPY!!! #NoWonder the name of this sho...	train
1867527	0x38959e	[]	In every circumtance I'd like to be thankful t...	train
1867528	0x2cbca6	[blessyou]	there's currently two girls walking around the...	train
1867533	0x24faed	[]	Ah, corporate life, where you can date <LH> us...	train
1867534	0x34be8c	[Sundayvibes]	Blessed to be living #Sundayvibes <LH>	train

1455563 rows × 4 columns

首先，我們從 Twitter 的貼文中提取留言和 hashtags，這些信息能夠有效捕捉到用戶的情緒和意圖。例如，hashtags 常用於表達情緒、主題或事件。為了準備情緒分析的數據，我們將留言和 hashtags 合併，形成一個可以進行情緒分類的文本。

1.2 BERT 嵌入

BERT (Bidirectional Encoder Representations from Transformers) 是一種強大的語言模型，能夠捕捉詞的上下文關係及其語義。選擇 bert-base-uncased 作為預訓練模型，因為該模型在多個 NLP 任務中已經表現出色。利用 BERT，我們將文本轉換為嵌入向量，作為 XGBoost 的輸入特徵。

```
# 將文字轉為 bert embeddings
def get_bert_embeddings(texts):
    embeddings = []
    for text in tqdm(texts):
        inputs = tokenizer(text, return_tensors='pt', padding=True, truncation=True, max_length=512)
        inputs = {key: value.to('cuda') for key, value in inputs.items()}  # 移動到 GPU
        with torch.no_grad():
            outputs = model(**inputs)
        cls_embedding = outputs.last_hidden_state[:, 0, :].cpu().numpy()  # 獲取 [CLS] token 的嵌入
        embeddings.append(cls_embedding)
    return np.vstack(embeddings)
```

Notebook editor cells

2. 使用 XGBoost 進行分類

在訓練 XGBoost 模型時使用以下參數：

1. max_depth:

- 說明：該參數控制樹的最大深度。較大的深度會導致模型更複雜，可能造成過擬合。

- 實驗結果：
    - 測試深度為 3, 5, 7 的模型，發現使用 max_depth=5 的模型在驗證集上性能最佳，因為此深度可以平衡模型的複雜性和泛化能力。

2. eta（學習率）

- 說明：控制每棵樹的貢獻，即每次更新的步伐大小。較小的學習率可以使模型學習更精細，但可能需要更多的樹來學習。
- 實驗結果：
    - 測試學習率為 0.01, 0.1 和 0.3，發現 eta=0.1 的模型表現最佳，因為它能夠達到較好的收斂速度，並且在驗證指標上表現穩定。

3. n_estimators（樹的棵數）

- 說明：該參數決定了要生成的樹的數量。更多的樹會增加模型的容量，但過多也會增加過擬合的風險。
- 實驗結果：
    - 嘗試 50, 100 和 200 棵樹，發現使用 n_estimators=100 能在驗證集上獲得最佳表現。

4. min_child_weight

- 說明：該參數控制葉子節點中最小的樣本權重和。進一步提高此值會減少過擬合的風險。
- 實驗結果：
    - 測試 min_child_weight 的值為 1 和 5，發現小的值（1）能讓模型更靈活，對於訓練數據的表現更好，但可能會在驗證集上過擬合。最終選擇 min_child_weight=1。

5. subsample 和 colsample_bytree

- 說明：這些參數控制訓練模型時用於隨機抽樣的訓練數據和特徵，這有助於提高模型的泛化能力。
- 實驗結果：
    - 對 subsample 和 colsample_bytree 測試了不同的比例（如 0.5, 0.8 和 1），發現使用 0.8 的值可以平衡模型的表現和計算效率。

3. 驗證

最後對驗證資料集的驗證結果如下：

ook editor cells

	precision	recall	f1-score	support
0	0.64	0.42	0.51	5052
1	0.47	0.87	0.61	10198
2	0.64	0.10	0.17	1274
3	0.44	0.36	0.39	3915
4	0.38	0.30	0.33	2713
5	0.57	0.15	0.24	4062
6	0.87	0.09	0.16	978
7	0.84	0.11	0.19	792
accuracy			0.49	28984
macro avg	0.61	0.30	0.33	28984
weighted avg	0.53	0.49	0.44	28984

4. 改進模型的建議

- **數據增強或補充**：檢查是否存在類別不平衡的情況。可以通過過採樣或欠採樣技術平衡類別。
- **模型調參**：進行交叉驗證，調整 XGBoost 的超參數，如增加樹的數量、調整學習率或樹的深度等。
- **使用不同的模型**：可以考慮集成學習或其他模型（如 LightGBM 或 CatBoost）來進行比較。
- **特徵工程**：檢查輸入特徵，是否有必要進行進一步的處理或增加其它信息來改進模型性能。