Kaggle Competition說明

Name : 鄭丞恩
Group : 83
2024/12/1

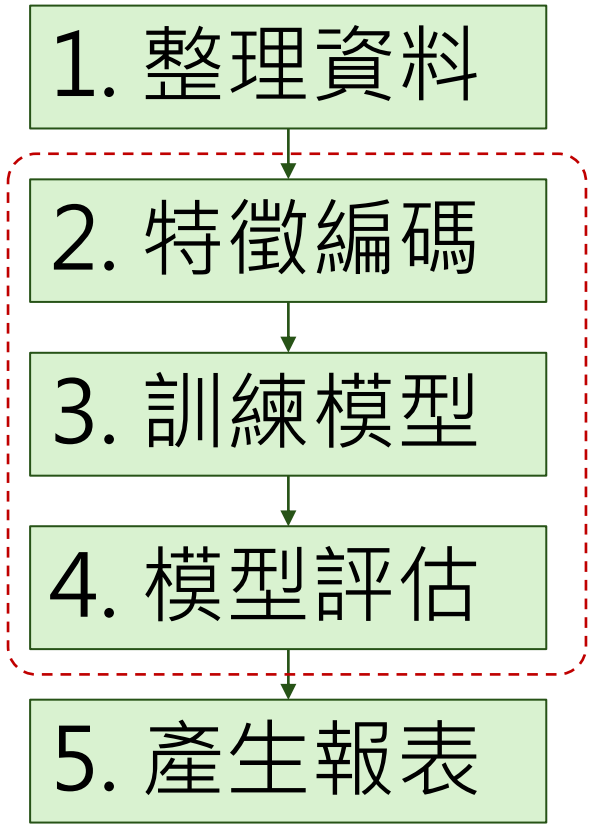
針對原始碼，寫下註解

python 複製程式碼

```python
# 引入必要的庫
import json
import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
from collections import Counter
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# ======== 資料讀取與整理 ========
# 1. 讀取 JSON 格式的推文數據
data = []
with open('/kaggle/input/dm-2024-isa-5810-lab-2-homework/tweets_DM.json', 'r') as f:
    for line in f:
        data.append(json.loads(line))
```

```python
# 4. 合併情緒標籤到訓練資料中
train_data = train_data.merge(emotion, on='tweet_id', how='left')

# 5. 移除訓練集中重複的文本
train_data.drop_duplicates(subset=['text'], keep=False, inplace=True)

# ======== 訓練數據準備 ========
# 1. 抽取部分訓練數據（30% 抽樣）
train_data_sample = train_data.sample(frac=0.3, random_state=42)

# 2. 提取目標變數（情緒）和特徵變數（文本）
y_train_data = train_data_sample['emotion']
X_train_data = train_data_sample.drop(['tweet_id', 'emotion', 'identification', 'hashtags'

# 3. 訓練集與測試集劃分（分層抽樣確保類別分佈一致）
X_train, X_test, y_train, y_test = train_test_split(
    X_train_data, y_train_data, test_size=0.2, random_state=42, stratify=y_train_data
)

# ======== 文本特徵轉換 ========
# 1. 使用 TF-IDF 將文本轉換為數值特徵
tfidf = TfidfVectorizer(max_features=500)
X = tfidf.fit_transform(X_train['text']).toarray()
X_test = tfidf.transform(X_test['text'])
```

萃取註解，重新了解程式結構



重點在這，有哪些選擇適合？

2. 特徵編碼

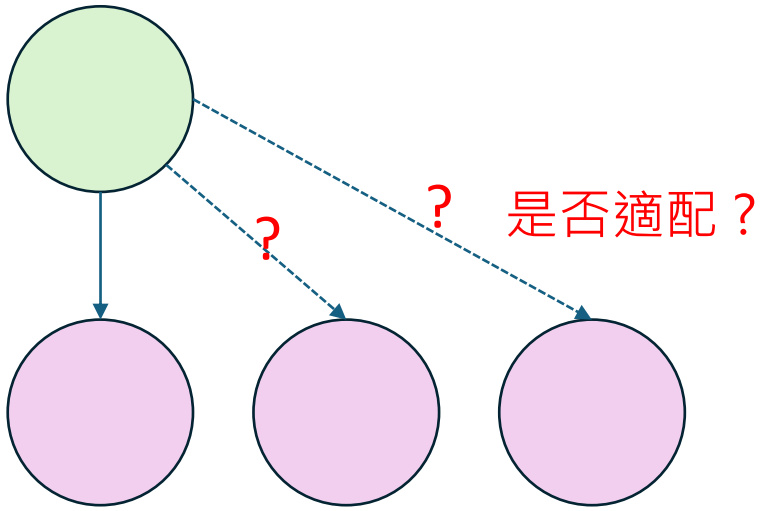
3. 訓練模型

→ 小組討論

怎樣管理(安排)還是得知道

2. 特徵編碼 → 可能有 n 個

3. 訓練模型 → 可能有 k 個



適配性？ 繼續討論

為何使用 TF-IDF

1. **計算詞語的重要性**
   - TF-IDF（Term Frequency-Inverse Document Frequency）是一種統計方法，能衡量單詞在文檔中的重要性。
   - **TF（詞頻）**：詞在文檔中出現的頻率，衡量詞在該文檔中的重要程度。
   - **IDF（逆文檔頻率）**：考慮詞在整個語料庫中的稀有程度，常見詞（例如 "the", "is"）權重降低，稀有詞權重提高。
2. **適合處理短文本數據**
   - 推文數據通常較短且具有稀疏性，TF-IDF 可以幫助突顯具有區別性的單詞，而非只關注常見詞。
3. **簡單且高效**
   - 生成的向量維度可以由 `max_features` 控制，計算相對高效，適合在基礎情感分析中作為文本表示。

為何非他不可？
編碼**為何要用TF-IDF**？

可替代方法

1. **詞嵌入方法**
   - 將詞轉換為固定維度的數值向量，保留詞之間的語義關係，效果通常優於 TF-IDF，適合更複雜的情感分析。

   (1) Word2Vec
   - 訓練或使用預訓練的詞嵌入模型（如 Google 的 Word2Vec）。
   - 缺點：對短文本表現有限，因為它忽略詞序與上下文關係。

組員要**解釋原因**
以及想出**可替代方法**

編碼方法問完後如法炮製
模型問題也問一遍

1. 傳統機器學習模型

(1) 支持向量機 (SVM)

- 特點
  - 適合高維稀疏數據（如 TF-IDF 向量）。
  - 通過選擇合適的核函數（如線性核或 RBF 核），能很好地處理非線性數據。
- 適用場景
  - 資料量中等，特徵維度較高的文本分類問題。
- 注意事項
  - 訓練時間較長，尤其是當樣本數量較多時。

(2) 邏輯回歸 (Logistic Regression)

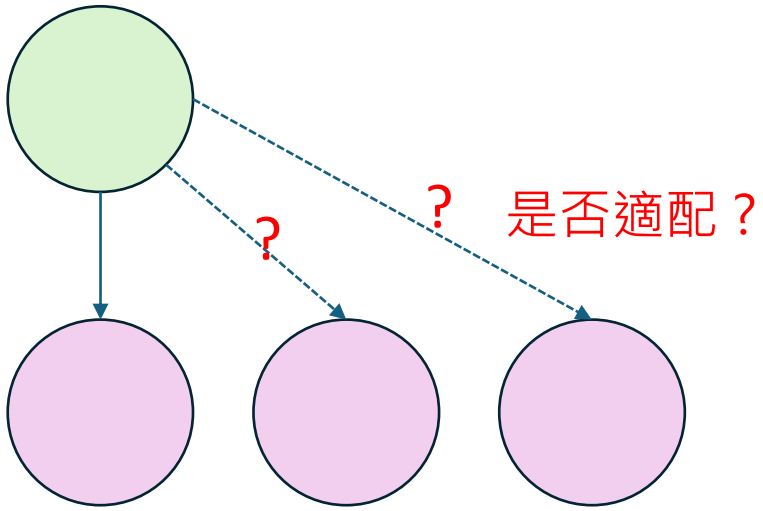
- 特點
  - 用於二分類或多分類任務，簡單高效，計算資源需求低。
  - 通過正則化項（L1 或 L2）可避免過擬合。
- 適用場景
  - 標籤數量較少（如二元情緒分類），[illegible]數據規模小。

為何非**隨機森林**不可？
模型除了**隨機森林**還有其他?
組員要**解釋原因**
以及列出**其他模型**

討論出一堆編碼和模型
身為組長還是要小心被騙

2. 特徵編碼 → 可能有 n 個

3. 訓練模型 → 可能有 k 個



特徵編碼 跟 訓練模型
之間是否有搭配問題？

→ 叫組員去整理表格出來

關鍵來了，要懂得問

- 假設依照上面介紹的內容，在相同資料的前提之下，特徵轉換 和 採用的模型 是否有特定的搭配關係？

假設依照上面介紹的內容，在相同資料的前提之下，特徵轉換 和 採用的模型 是否有特定的搭配關係？

關鍵問題

Bingo !

是的 在特徵轉換和採用模型之間，確實存在一定的搭配關係。不同的特徵表示方式可能會影響模型的效果，因為每種模型對數據的需求與表現方式有所不同。以下根據特徵轉換與模型的匹配性進行分析：

---

1. 傳統特徵轉換方法與模型搭配

(1) TF-IDF

- 適合模型：
  - 支持向量機 (SVM)：高維稀疏數據表現良好，尤其是線性分類器。
  - 隨機森林 (Random Forest)：對特徵選擇不敏感，但在高維特徵下仍表現穩定。
  - 梯度提升樹 (如 XGBoost, LightGBM)：適合處理稀疏特徵數據，且能自動選擇重要特徵。
  - 邏輯回歸 (Logistic Regression)：適用於線性分類，對高維數據有不錯的效果。
- 不適合模型：
  - 深度學習模型（如 CNN, RNN, Transformer）：這些模型需要更語義化的特徵表示（如詞嵌入），TF-IDF 無法捕捉上下文信息。

(2) Bag of Words (詞袋模型)

- 適合模型：
  - 與 TF-IDF 相似，適合 SVM、隨機森林、邏輯回歸。
  - 特徵簡單，對文本語序和語義無要求，適合基礎任務有效。

模型與特徵轉換的搭配總結

模型與特徵轉換的搭配總結

模型	推薦特徵轉換	不推薦特徵轉換	適用場景
隨機森林 (Random Forest)	TF-IDF, Bag of Words, Word2Vec 平均向量	Contextual Embedding (需降維處理)	稀疏或非線性特徵數據，模型穩定性需求高。
支持向量機 (SVM)	TF-IDF, Bag of Words, Word2Vec 平均向量	One-Hot Encoding, Raw Text	高維稀疏特徵，分類邊界清晰。
邏輯回歸 (Logistic Regression)	TF-IDF, Bag of Words, One-Hot Encoding	Contextual Embedding	特徵數據規模小，線性分類需求。
梯度提升樹 (XGBoost, LightGBM)	TF-IDF, Word2Vec 平均向量	Contextual Embedding (需降維處理)	特徵豐富、需要高精準度的分類或迴歸任務。
Bagging (如隨機森林)	TF-IDF, Bag of Words, Word2Vec 平均向量	Contextual Embedding	適合高維稀疏數據，降低模型方差。
Boosting (XGBoost, LightGBM)	TF-IDF, Word2Vec 平均向量	Contextual Embedding	特徵豐富且密集，對精準度需求高的分類任務。
AdaBoost	TF-IDF, Bag of Words, Word2Vec 平均向量	Contextual Embedding	適合小型數據集，對難分類樣本有強適應性。
卷積神經網絡 (CNN)	Word2Vec, FastText, GloVe	TF-IDF, Bag of Words	文本局部特徵提取，適合短文本或固定長度文本。
循環神經網絡 (RNN, LSTM)	Word2Vec, GloVe, FastText, Contextual Embedding	TF-IDF, Bag of Words	捕捉上下文關係，處理長文本和序列數據。
Transformer (BERT)	Contextual Embedding	TF-IDF, Bag of Words	高語義需求的情緒分類或上下文語義理解任務。

還等甚麼？
請組員把這**搭配組合**做表格出來

搭配組合配方出來了，接下來就是生產問題

生產最重視：**效率**跟**成本**

關鍵生產要素：效率跟成本

- 效率：配方(編碼+模型) 的 → 準確性Accuracy
- 成本：最快生產完成的關鍵條件 → 用GPU能否加速

於是我們寫了六支程式

- 1. TF-IDF + Randon_forest (助教提供 , 有稍微改一些)
- 2. TF-IDF + Boosting (XGBoost) w/GPU
- 3. Word2Vec + 隨機森林
- 4. Word2Vec + CNN w/GPU
- 5. BERT 嵌入 + Transformer w/GPU
- 6. LSTM-Tensorflow-Tokenizer

- Github link : https://github.com/EnzoCheng222/DM2024-Lab2-Master/tree/main/3.%20Kaggle_Competition

寫程式(生產)之前就把期望值預估好

效率預估期望值

方法	特徵工程	模型	準確率預估	優勢	劣勢	GPU 支援
TF-IDF + 隨機森林	稀疏特徵表示，詞頻與逆文檔頻率權重	隨機森林	75%-82%	模型穩定性強，對噪聲和高維數據不敏感	無法處理非線性模式，對語義信息利用不足	不支持
TF-IDF + Boosting	稀疏特徵表示，詞頻與逆文檔頻率權重	XGBoost 或 LightGBM	78%-85%	擅長處理稀疏特徵，對錯分樣本有良好適應能力	訓練成本略高，需調參以達到最佳效果	支持（顯著加速，適合大數據集）
Word2Vec + 隨機森林	詞嵌入，計算句向量平均值	隨機森林	72%-80%	能結合詞嵌入語義特徵，提升語義捕捉能力	詞嵌入需預處理，隨機森林對非線性語義的處理有限	不支持
Word2Vec + CNN	詞嵌入，保留語序	卷積神經網絡	75%-85%	捕捉局部語義特徵，對短文本效果佳	訓練需較多資源，對長文本效果有限	支持（顯著加速）
BERT 嵌入 + Transformer	上下文語義嵌入，保留全局語義	預訓練 BERT 模型	85%-90%	能捕捉上下文語義，分類準確率最高	訓練和推理成本高，需要大量數據和資源支持	支持（必要，否則速度較慢）
Tokenizer + LSTM	數字化文本序列，保留序列上下文	長短期記憶神經網絡	80%-88%	能捕捉文本序列特徵，適合時間序列或長文本	訓練成本中等，對長文本可能有梯度消失問題	支持（顯著加速，必要）

成本預估期望值

整理好全部方法資料
準確率、使用範圍(優/劣勢)、GPU支援情況

完成程式碼撰寫(持續測試中)

準確率紀錄：

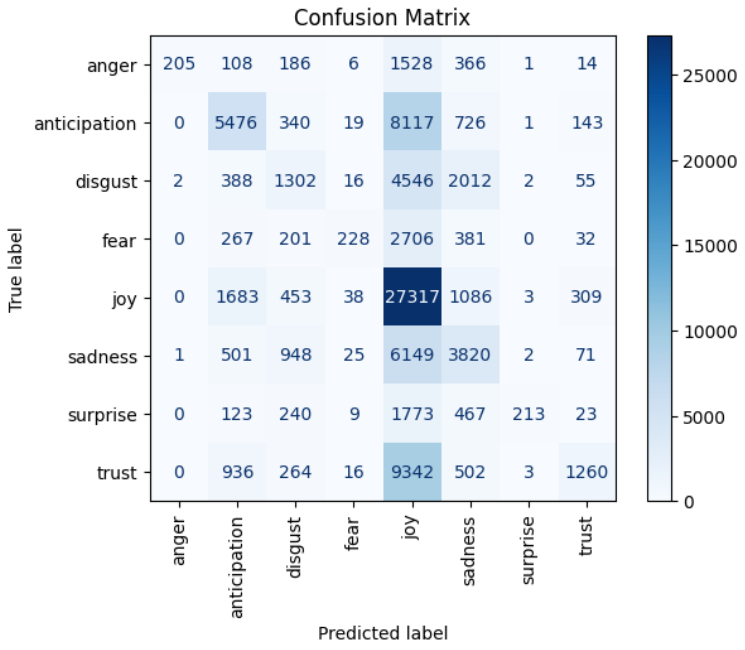
- Accuracy: 0.4586
- Accuracy: 0.4580

3. Word2Vec + 隨機森林 Draft saved

File Edit View Run Settings Add-ons Help

Run All Code

```python
# 引入必要的庫
import json
import pandas as pd
import numpy as np
import nltk
from nltk.tokenize import word_tokenize
from gensim.models import Word2Vec
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# 確保 NLTK 資源可用
nltk.download('punkt')

# ======== 資料讀取與整理 ========
# 1. 讀取 JSON 格式的推文數據
data = []
with open('/kaggle/input/dm-2024-isa-5810-lab-2-homework/tweets_DM.json', 'r') as f:
    for line in f:
        data.append(json.loads(line))
f.close()

# 2. 讀取其他 CSV 文件
emotion = pd.read_csv('/kaggle/input/dm-2024-isa-5810-lab-2-homework/emotion.csv')
data_identification = pd.read_csv('/kaggle/input/dm-2024-isa-5810-lab-2-homework/data_identification.csv')

# ======== 數據處理 ========
# 1. 從 JSON 數據提取 tweet_id, hashtags, 和文本內容
df = pd.DataFrame(data)
_source = df['_source'].apply(lambda x: x['tweet'])
df = pd.DataFrame({
    'tweet_id': _source.apply(lambda x: x['tweet_id']),
    'hashtags': _source.apply(lambda x: x['hashtags']),
    'text': _source.apply(lambda x: x['text']),
})

# 2. 合併資料集標籤 (train/test)
df = df.merge(data_identification, on='tweet_id', how='left')
```




```
----

Confusion Matrix:
[[  205   108   186     6  1528   366     1    14]
 [    0  5476   340    19  8117   726     1   143]
 [    2   388  1302    16  4546  2012     2    55]
 [    0   267   201   228  2706   381     0    32]
 [    0  1683   453    38 27317  1086     3   309]
 [    1   501   948    25  6149  3820     2    71]
 [    0   123   240     9  1773   467   213    23]
 [    0   936   264    16  9342   502     3  1260]]

----
```

解決造成模型準確率偏低的雜訊問題

執行程式碼之後結果比較

方法	特徵工程	模型	準確率預估	採用原始資料	GPU 支援
TF-IDF + 隨機森林	稀疏特徵表示，詞頻與逆文檔頻率權重	隨機森林	75%-82%	0.4808	No
TF-IDF + Boosting	稀疏特徵表示，詞頻與逆文檔頻率權重	XGBoost 或 LightGBM	78%-85%	0.4780	GPU T4 x 2
Word2Vec + 隨機森林	詞嵌入，計算句向量平均值	隨機森林	72%-80%	0.4586	No
Word2Vec + CNN	詞嵌入，保留語序	卷積神經網絡	75%-85%	0.5514	GPU T4 x 2
BERT 嵌入 + Transformer	上下文語義嵌入，保留全局語義	預訓練 BERT 模型	85%-90%	0.6439	GPU T4 x 2

ノイズ

Noise

雜訊

ノイズ
Noise
雜訊

誤判為雜訊 → → 加強訊號

Recycle it！

確定是雜訊 →

Just Delete it！

看一下 data

emojis (絵文字)

Tag : <LH>

```python
import json

data = []
with open('/kaggle/input/dm-2024-isa-5810-lab-2-homework/tweets_DM.json', 'r') as f:
    for line in f:
        data.append(json.loads(line))

# 獲取前 2000 筆資料
first_2000_entries = data[:2000]

# 列印前 2000 筆資料
for entry in first_2000_entries:
    print(entry)
```


```
{'_score': 391, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': ['Snapchat'], 'tweet_id': '0x376b20', 'text': 'People who post "add me on #Snapchat" must be dehydrated. Cuz man.... that\'s <LH>'}}, '_crawl
date': '2015-05-23 11:42:47', '_type': 'tweets'}
{'_score': 433, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': ['freepress', 'TrumpLegacy', 'CNN'], 'tweet_id': '0x2d5350', 'text': '@brianklaas As we see, Trump is dangerous to #freepress around the worl
d. What a <LH> <LH> #TrumpLegacy.  #CNN'}}, '_crawldate': '2016-01-28 04:52:09', '_type': 'tweets'}
{'_score': 232, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': ['bibleverse'], 'tweet_id': '0x28b412', 'text': 'Confident of your obedience, I write to you, knowing that you will do even more than I ask.
(Philemon 1:21) 3/4 #bibleverse <LH> <LH>'}}, '_crawldate': '2017-12-25 04:39:20', '_type': 'tweets'}
{'_score': 376, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': [], 'tweet_id': '0x1cd5b0', 'text': 'Now ISSA is stalking Tasha 😩😩😩 <LH>'}}, '_crawldate': '2016-01-24 23:53:05', '_type': 'tweets'}
{'_score': 989, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': [], 'tweet_id': '0x2de201', text': '"Trust is not the same as faith. A friend is someone you trust. Putting faith in anyone is a mistake." ~
Christopher Hitchens <LH> <LH>'}}, '_crawldate': '2016-01-08 17:18:59', '_type': 'tweets'}
{'_score': 120, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': ['authentic', 'LaughOutLoud'], 'tweet_id': '0x1d755c', 'text': '@RISKshow @TheKevinAllison Thx for the BEST TIME tonight. What stories! Heart
breakingly <LH> #authentic #LaughOutLoud good!!'}}, '_crawldate': '2015-06-11 04:44:05', '_type': 'tweets'}
{'_score': 1021, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': [], 'tweet_id': '0x2c91a8', 'text': 'Still waiting on those supplies Liscus. <LH>'}}, '_crawldate': '2015-08-18 02:30:07', '_type': 'tweet
s'}
{'_score': 481, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': [], 'tweet_id': '0x368e95', 'text': 'Love knows no gender. 😢😭 <LH>'}}, '_crawldate': '2015-08-20 14:31:27', '_type': 'tweets'}
{'_score': 827, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': ['LeagueCup'], 'tweet_id': '0x249c0c', 'text': '@DStvNgCare @DStvNg More highlights are being shown than actual sports! Who watches triathlon
highlights anyway? <LH> #LeagueCup'}}, '_crawldate': '2016-04-18 13:01:02', '_type': 'tweets'}
{'_score': 66, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': ['materialism', 'money', 'possessions'], 'tweet_id': '0x218443', 'text': 'When do you have enough ? When are you satisfied ? Is you goal reall
y all about money ?  #materialism #money #possessions <LH>'}}, '_crawldate': '2015-09-09 09:22:55', '_type': 'tweets'}
{'_score': 631, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': ['SSM', 'gender', 'diversity'], 'tweet_id': '0x359db9', 'text': 'The #SSM debate; <LH> (a manufactured fantasy used to distract the ignorant
masses from their mundane lives) V #gender #diversity (a m.....'}}, '_crawldate': '2016-08-05 21:15:36', '_type': 'tweets'}
{'_score': 839, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': [], 'tweet_id': '0x23b037', 'text': "I love suffering 🙃🙃 I love when valium does nothing to help 🙃🙃 I love when my doctors say that the
y've done all they can 🙃🙃 <LH>"}}, '_crawldate': '2016-01-28 08:11:22', '_type': 'tweets'}
{'_score': 560, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': ['Pissed'], 'tweet_id': '0x1fde89', 'text': 'Can someone tell my why my feeds scroll back to the same 30 tweets that I saw 1 min ago? #Pisse
d!'}}, '_crawldate': '2016-01-26 22:09:45', '_type': 'tweets'}
{'_score': 887, '_index': 'hashtag_tweets', '_source': {'tweet': {'hashtags': ['justgradstudentthings', 'ecology'], 'tweet_id': '0x37a0a9', 'text': 'You know you research butterflies when predictive text autocorrects "b
ut" to "butterfly" #justgradstudentthings #ecology <LH>'}}, '_crawldate': '2015-08-11 16:22:15', '_type': 'tweets'}
```

我又寫了程式：查html tag數量

```python
import json
from collections import Counter
from bs4 import BeautifulSoup
import pandas as pd

# 初始化資料列表
data = []

# 讀取 JSON 文件
with open('/kaggle/input/dm-2024-isa-5810-lab-2-homework/tweets_DM.json', 'r') as f:
    for line in f:
        data.append(json.loads(line))

# 初始化計數器
tag_counter = Counter()

# 遍歷每條推文
for entry in data:
    tweet_text = entry['_source']['tweet']['text']
    # 使用 BeautifulSoup 解析推文文本
    soup = BeautifulSoup(tweet_text, 'html.parser')
    # 提取所有標籤
    tags = [tag.name for tag in soup.find_all()]
    # 更新計數器
    tag_counter.update(tags)

# 將計數結果轉換為 DataFrame
df_tags = pd.DataFrame(tag_counter.items(), columns=['Tag', 'Count'])

# 根據出現次數進行排序
df_tags = df_tags.sort_values(by='Count', ascending=False).reset_index(drop=True)

# 顯示結果
# print(df_tags)
```

```python
print(df_tags)
```

	Tag	Count
0	lh	2296492

搞半天只有 <lh>
但數量有 2,296,492 個

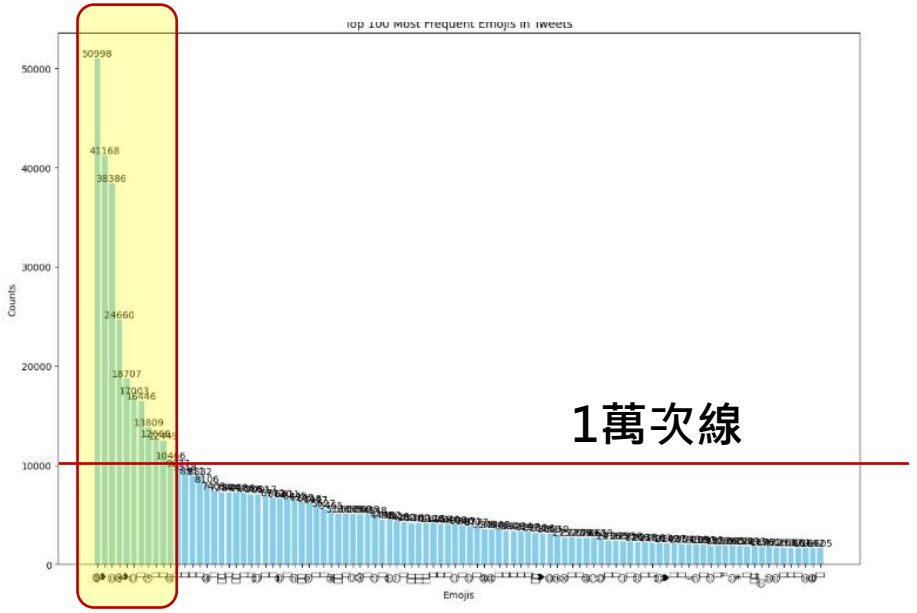
沒用

Just Delete it！

我又寫了程式：查emoji 數量 & 分布

```python
import json
import regex
import collections
import matplotlib.pyplot as plt

# 定義從文本中提取表情符號的函數
def extract_emojis(text):
    """從給定的文本中提取所有表情符號。"""
    emoji_list = []
    data = regex.findall(r'\X', text)
    for word in data:
        if any(char in emoji.EMOJI_DATA for char in word):
            emoji_list.append(word)
    return emoji_list

# 讀取 JSON 檔案並解析為列表
data = []
with open('/kaggle/input/dm-2024-isa-5810-lab-2-homework/tweets_DM.json', 'r') as f:
    for line in f:
        data.append(json.loads(line))

# 提取所有推文中的表情符號
all_emojis = []
for entry in data:
    tweet_text = entry['_source']['tweet']['text']
    emojis_in_tweet = extract_emojis(tweet_text)
    all_emojis.extend(emojis_in_tweet)

# 計算每個表情符號的出現次數
emoji_counter = collections.Counter(all_emojis)

# 取得出現頻率最高的前 100 個表情符號
top_500_emojis = emoji_counter.most_common(500)

# 分離表情符號和對應的計數
emojis, counts = zip(*top_500_emojis)

# 設定圖表大小
plt.figure(figsize=(15, 10))

# 繪製直方圖
```

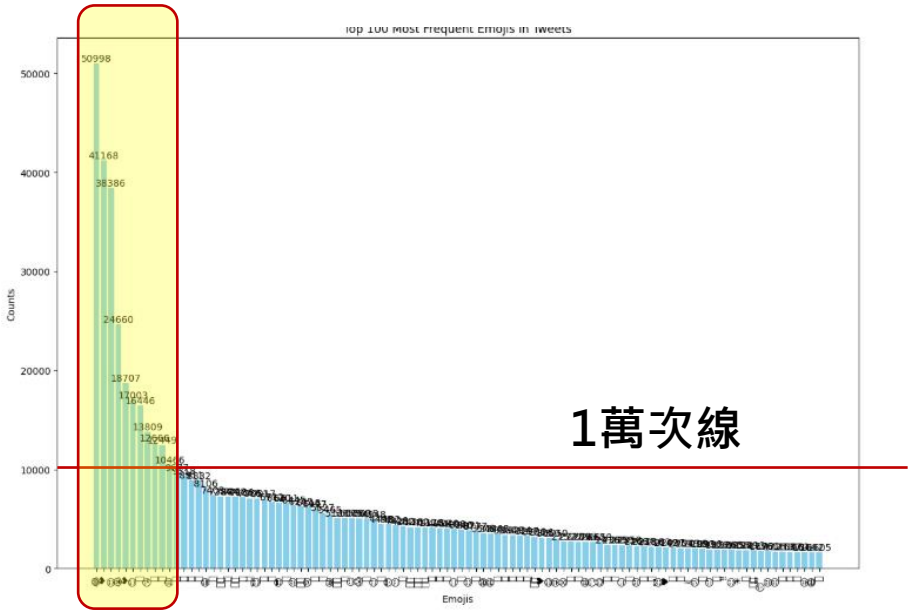
```python
[7]: top_500_emojis

[7]: [('😂', 50998),
      ('❤️', 41168),
      ('😍', 38386),
      ('😭', 24660),
      ('❤', 18707),
      ('😊', 17003),
      ('🙏', 16446),
      ('😘', 13809),
      ('💕', 12666),
      ('🔥', 12449),
      ('😩', 10466),
      ('😉', 9677),
      ('💯', 9318),
      ('💙', 8911),
      ('😁', 8832),
      ('😜', 8106),
      ('👬', 7403),
      ('🙏', 7284),
      ('👍', 7284),
      ('🙏', 7262),
      ('✨', 7186),
      ('💜', 7086),
      ('😇', 7051),
      ('👏', 7017),
      ('😋', 6712),
      ('😎', 6620),
      ('💔', 6611),
```



高頻群體
高頻使用這些emoji可以解釋為有明顯情感
在模型當中可以視為具有明顯特徵

Recycle it！

高頻群體處理方式



高頻群體
高頻使用這些emoji可以解釋為有明顯情感
在模型當中可以視為具有明顯特徵

Recycle it !

加強訊號

讓特徵更明顯

特徵強化&保留

讓特徵更明顯

特徵強化

Emoji	英文含義	單詞
😂	Face with Tears of Joy	[joy]
💗	Red Heart	[love]
😍	Smiling Face with Heart-Eyes	[love]
😭	Loudly Crying Face	[cry]
💗	Red Heart	[love]
😊	Smiling Face with Smiling Eyes	[happy]
🙏	Folded Hands	[pray]
😘	Face Blowing a Kiss	[kiss]
💕	Two Hearts	[love]
🔥	Fire	[fire]
😩	Weary Face	[weary]
🤔	Thinking Face	[think]
💯	Hundred Points	[perfect]
💙	Blue Heart	[loyalty]
🙄	Face with Rolling Eyes	[annoyed]
😁	Beaming Face with Smiling Eyes	[happy]
🙌	Raising Hands	[celebrate]
🙏	Folded Hands: Medium-Dark Skin Tone	[pray]
👍	Thumbs Up	[approve]
🙏	Folded Hands: Medium Skin Tone	[pray]

做成字典

- 把超過1萬次左右的emoji納入字典，當作過濾器做轉換用途。

特徵保留

```
emoji_dict = {
    '😂': '[joy]',
    '❤️': '[love]',
    '😍': '[love]',
    '😭': '[cry]',
    '💚': '[love]',
    '😊': '[happy]',
    '🙏': '[pray]',
    '😘': '[kiss]',
    '💖': '[love]',
    '🔥': '[fire]',
    '😩': '[weary]',
    '🤔': '[think]',
    '💯': '[perfect]',
    '💙': '[love]',
    '🙄': '[annoyed]',
    '😁': '[happy]',
    '🙌': '[celebrate]',
    '🙏': '[pray]',
    '👍': '[approve]',
    '🙏': '[pray]'
}
```

結果...

方法	特徵工程	模型	準確率預估	採用原始資料	雜訊過濾後	GPU 支援
TF-IDF + 隨機森林	稀疏特徵表示，詞頻與逆文檔頻率權重	隨機森林	75%-82%	0.4808	0.4708	No
TF-IDF + Boosting	稀疏特徵表示，詞頻與逆文檔頻率權重	XGBoost 或 LightGBM	78%-85%	0.4780	0.4675	GPU T4 x 2
Word2Vec + 隨機森林	詞嵌入，計算句向量平均值	隨機森林	72%-80%	0.4586	0.4566	No
Word2Vec + CNN	詞嵌入，保留語序	卷積神經網絡	75%-85%	0.5514	0.5312	GPU T4 x 2
BERT 嵌入 + Transformer	上下文語義嵌入，保留全局語義	預訓練 BERT 模型	85%-90%	0.6439	0.6294	GPU T4 x 2

目前最好成績

Submissions

Select up to 2 submissions that will count towards your final leaderboard score. If less than 2 are selected, Kaggle will automatically select from your best scoring submissions. Learn More

0/2

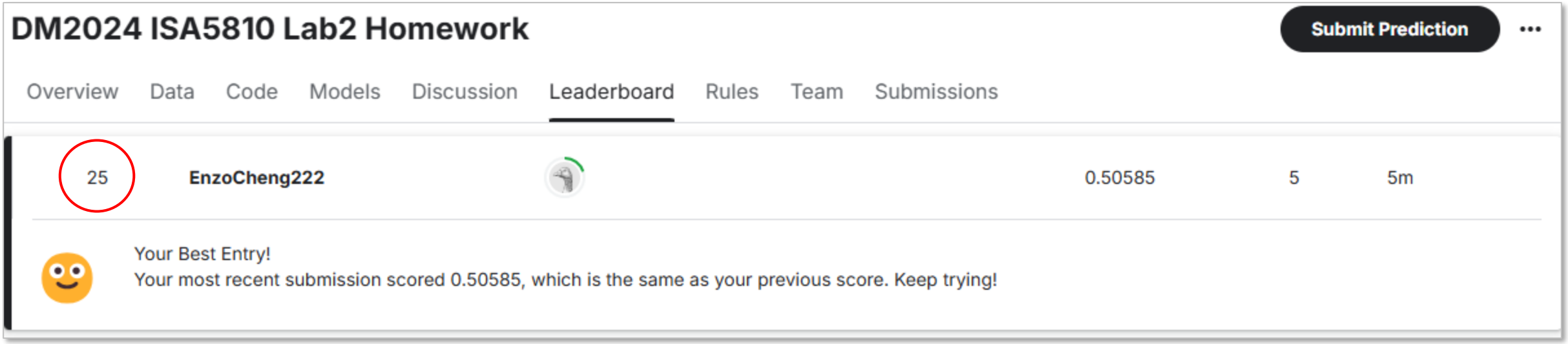
Auto-selection candidates

All | Successful | Selected | Errors

Recent

Submission and Description	Public Score	Select
BERT Transformer epoch 4.csv Complete · 1m ago · BERT + Transformer + epoch 4	0.50585	☐
submission(1).csv Complete · 2m ago · BERT + Transformer + epoch 4	0.50585	☐
1. TF-IDF + Random_forest - Version 5 Complete · 5d ago · Notebook 1. TF-IDF + Random_forest | Version 5	0.34984	☐
4. Word2Vec + CNN (GPU) - Version 2 Complete · 6d ago · Notebook 4. Word2Vec + CNN (GPU) | Version 2	0.40157	☐
1. TF-IDF + Random_forest - Version 3 Complete · 7d ago · Notebook 1. TF-IDF + Random_forest | Version 3	0.31825	☐

目前最佳排名：25！


DM2024 ISA5810 Lab2 Homework
Submit Prediction
Overview
Data
Code
Models
Discussion
Leaderboard
Rules
Team
Submissions
25
EnzoCheng222
0.50585
5
5m
Your Best Entry!
Your most recent submission scored 0.50585, which is the same as your previous score. Keep trying!