

我嘗試了兩種方法(檔案為 kaggle_v1 和 kaggle_v2)，第一個是手動調整參數觀察 macro 的 f1 score，使用 tfidf 將文字轉成向量，並使用隨機森林分類；第二個則是同樣的前處理，但使用隨機森林和 xgboost 的分類器，並配合 GridSearchCV 自動調整參數。

首先介紹一下第一種方法，也是期限內嘗試出最好的結果，以下可以看到其中 import 的 function。

```
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.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
```

將 text 列空白的去除

```
train_data = train_data.merge(emotion, on='tweet_id', how='left') # Merge emotion for corresponding tweet_id
train_data.drop_duplicates(subset=['text'], keep=False, inplace=True) # Remove duplication
```

一開始只有 sample0.1，後來發現 0.2 的效果有好一些。

```
train_data_sample = train_data.sample(frac=0.2) # Get sample
```

我採取將 hashtags 和 text 合成訓練資料的方式，這麼做的原因是因為我覺得 hashtags 的空值太多，透過合成一組訓練資料的方式簡化。

```
y_train_data = train_data_sample['emotion']
X_train_data = train_data_sample.drop(['tweet_id', 'emotion', 'identification'], axis=1)
X_train_data = X_train_data['text'] + ' ' + X_train_data['hashtags'].apply(lambda x: ' '.join(x)) # 將hashtags和text合成
X_train_data
```

分割訓練集和測試集，比例為 0.2

```
from sklearn.model_selection import train_test_split

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
)
# 將資料分成訓練和測試，比例為0,2

✓ 0.1s
```

TFIDF 最大特徵取 5000

```
tfidf = TfidfVectorizer(max_features=5000) # 使用TFIDF處理，最大特徵取5000
X = tfidf.fit_transform(X_train).toarray()
X_test = tfidf.transform(X_test)
```

3.3s

標記資料

```
le = LabelEncoder() # 標記轉成數字減少資料量
y = le.fit_transform(y_train)
y_test = le.transform(y_test)
```

0.0s

使用隨機森林分類並預測

```
rf_model = RandomForestClassifier() #使用隨機森林分類
model = rf_model.fit(X, y)
```

2m 49.5s

```
y_pred = model.predict(X_test) # 預測
```

查看指標，用來手動調參

```
# 查看各種指標
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score
print(accuracy_score(y_test, y_pred))
print(recall_score(y_test, y_pred, average='macro'))
print(precision_score(y_test, y_pred, average='macro'))
print(f1_score(y_test, y_pred, average='macro'))
```

讀取 test data 並用前面訓練的模型預測，轉成符合規定的.csv 檔

```
test_data = df[df['identification'] == 'test'] # 讀取test data
# test_data.to_csv('dm-2024-isa-5810-lab-2-homework/test000.csv', index=False)

# 和訓練集一樣的前處理
X_test_data = test_data.drop(['tweet_id', 'identification'], axis=1)
X_test_data['text'] = X_test_data['text'] + ' ' + X_test_data['hashtags'].apply(lambda x: ' '.join(x))










X_test_data = tfidf.transform(X_test_data).toarray()
y_test_pred = model.predict(X_test_data)
y_pred_labels = le.inverse_transform(y_test_pred)
submission = pd.DataFrame({
    'id': test_data['tweet_id'],
    'emotion': y_pred_labels
})
submission.to_csv('C:/DATAMINING_LAB2/dm-2024-isa-5810-lab-2-homework/submission_kaggle_v1.csv', index=False)
```

第二部分使用 GridSearchCV 最佳化隨機森林和 XGBoost，但似乎都出現過擬合的狀況 (因為要提交兩個 github 所以在轉移程式碼過程中似乎沒存到新檔案，所以 output 不太完整)。可以看到自己計算的 fi score 是算不錯的，但繳交後卻會降到 0.17 多，因此推測是過擬合。

```
warnings.warn(  
最佳參數: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 200, 'subsample': 0.8}  
最佳分數: 0.38790252460258295  
0.5086254485233233  
0.3375609739467981  
0.6766727143309054  
0.3959437386101053
```

```
Fitting 3 folds for each of 2 candidates, totalling 6 fits  
最佳參數: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 200, 'subsample': 0.5}  
最佳分數: 0.3887899426018993  
0.5105575489925476  
0.33910297554903746  
0.6723489341806004  
0.396659207824949
```

見下圖，高於 0.3 的都是使用隨機森林，而最高的分數模型是使用 kaggle_v1 中的參數。

 submission7.csv Complete (after deadline) · 18s ago	0.18570	0.17822	<input type="checkbox"/>
 submission6.csv Complete (after deadline) · 12m ago	0.18429	0.17981	<input type="checkbox"/>
 submission5.csv Complete (after deadline) · 1h ago	0.17897	0.17691	<input type="checkbox"/>
 submission_kaggle_v1.csv Complete (after deadline) · 5h ago	0.32620	0.34361	<input type="checkbox"/>
 submission4.csv Complete (after deadline) · 9h ago	0.17763	0.17358	<input type="checkbox"/>
 submission3.csv Complete (after deadline) · 3d ago	0.32802	0.34538	<input type="checkbox"/>
 submission2.csv Complete (after deadline) · 3d ago	0.32858	0.34089	<input type="checkbox"/>
 submission1.csv Complete · 4d ago	0.31260	0.32917	<input type="checkbox"/>
 submission.csv Complete · 4d ago	0.31527	0.33038	<input type="checkbox"/>

以下是兩種模型的最佳化部分程式碼

```
# 隨機森林最佳化
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# 使用 GridSearchCV 尋找最佳參數
xgb_clf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(xgb_clf, param_grid, cv=3, scoring='f1_macro', verbose=2, n_jobs=-1)
grid_search.fit(vec_X, vec_y)

# 最佳參數
print("最佳參數:", grid_search.best_params_)
print("最佳分數:", grid_search.best_score_)

best_model = grid_search.best_estimator_

# 使用最佳模型對測試數據進行預測
y_pred = best_model.predict(vec_X_val)
```

```
# 使用xgboost分類器
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV

param_grid = {
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'n_estimators': [50, 100, 150]
}

# 使用 GridSearchCV 尋找最佳參數
xgb_clf = XGBClassifier(tree_method='hist', device='cuda', random_state=42, objective='multi:softmax', num_class=8)
grid_search = GridSearchCV(xgb_clf, param_grid, cv=3, scoring='f1_macro', verbose=2, n_jobs=-1)
grid_search.fit(vec_X, vec_y)

# 最佳參數
print("最佳參數:", grid_search.best_params_)
print("最佳分數:", grid_search.best_score_)

best_model = grid_search.best_estimator_

# 使用最佳模型對測試數據進行預測
y_pred = best_model.predict(vec_X_val)
```