113033597 譚嘉葳

1. 導入必要的庫
   - 數據處理和分析：pandas 和 numpy 用於數據操作和計算。
   - 文本處理：TfidfVectorizer 將文本轉換為特徵向量。
   - 機器學習模型：RandomForestClassifier 是主要分類算法，用於情緒預測。
   - 評估：classification_report 提供模型的詳細評估指標（如精確率、召回率）。

```
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
```

2. 加載數據
   - tweets_DM.json：這是包含推文的原始 JSON 文件。每行是 JSON 格式的推文數據。
   - emotion.csv：這是一個 CSV 文件，存有每條推文的情緒標籤（如喜悅、悲傷等）。
   - data_identification.csv：這是另一個 CSV 文件，標記每條推文是訓練數據還是測試數據。

```
# Read data

data = []
with open('/Users/vivian/OneDrive - NTHU/文件/清大/Data mining/DMLab2/hw/tweets_DM.json', 'r') as f:
    for line in f:
        data.append(json.loads(line))

f.close()
emotion = pd.read_csv('C:/Users/vivian/OneDrive - NTHU/文件/清大/Data mining/DMLab2/hw/emotion.csv')
data_identification = pd.read_csv('C:/Users/vivian/OneDrive - NTHU/文件/清大/Data mining/DMLab2/hw/data_identification.csv')
```

3. 創建新 DataFrame
   - 提取 JSON 文件中的有用字段：
   - tweet_id：推文的唯一標識符。
   - hashtags：推文中的標籤，用於增強語義信息。
   - text：推文的文本內容。
   - 合併數據：根據 tweet_id 將 data_identification.csv 中的標記合併到主 DataFrame 中，用於分離訓練集和測試集。

```python
# Filter out the 'train' data
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']),
})
df = df.merge(data_identification, on='tweet_id', how='left')

train_data = df[df['identification'] == 'train']
```

4. 數據預處理
   - 分離訓練數據：僅保留 identification 為 train 的數據。
   - 合併情緒標籤：將情緒數據（emotion.csv）與訓練數據合併，為每條推文分配情緒標籤。
   - 移除重複推文：避免同一條推文對模型的訓練造成偏差。

```python
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
```

```python
train_data_sample = train_data.sample(frac=0.1) # Get sample
```



```python
train_data_sample
```

5. 特徵工程
   - 目標變量：y_train_data 是情緒標籤列。
   - 特徵選擇：移除與模型無關的列（tweet_id 和 identification）。
   - 合併文本與標籤：將推文的 text 和 hashtags 合併，形成完整的語義表達。

```python
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)) # Combine text and hashtags
```

6. 訓練數據分割
   - 數據拆分：80% 作為訓練集，20% 作為測試集。
   - 類別平衡：stratify 確保拆分後的數據中，各情緒類別的分佈與原始數據一致。

```python
from sklearn.model_selection import train_test_split
# Split training and testing data for evaluation.
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)
```

7. 文本向量化
   - TF-IDF：將文本轉換為特徵矩陣，表示每個詞的重要性。此處僅保留 1000 個最具代表性的詞。

- 數據轉換：fit_transform 用於訓練數據，transform 用於測試數據。

```
tfidf = TfidfVectorizer(max_features=1000) # Use tfidfVectorizer and remove stop_words.
X = tfidf.fit_transform(X_train).toarray()
X_test = tfidf.transform(X_test)
```

8. 標籤編碼
   - 將情緒標籤（如 "joy", "sadness"）編碼為數字類別（如 0, 1）。

```
le = LabelEncoder() # Label target
y = le.fit_transform(y_train)
y_test = le.transform(y_test)
```

9. 模型訓練和預測
   - 訓練模型：隨機森林模型學習訓練數據的特徵與標籤的關係。
   - 進行預測：模型在測試數據集上進行分類。

```
clf = RandomForestClassifier() # Use RandomForest model
clf.fit(X, y)
model = clf



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

10. 評估模型
    - 計算模型的準確率（Accuracy），即正確分類的比例。

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred) # Evaluation
```

11. 測試數據處理
    - 清理測試數據：類似訓練數據的處理，將文本與標籤合併。
    - 向量化：使用相同的 TF-IDF 模型轉換測試數據。

```
# Do the same thing as training stage, but here we don't have emotions feature.
X_test_data = test_data.drop(['tweet_id', 'identification'], axis=1)
X_test_data = X_test_data['text'] + ' ' + X_test_data['hashtags'].apply(lambda x: ' '.join(x))



X_test_data = tfidf.transform(X_test_data).toarray() # Convert test data by using same tfidfVectorizer
```

12. 輸出結果

- 預測情緒：模型對測試數據進行情緒分類。
- 標籤反轉：將編碼標籤轉換回原始的情緒名稱。

```
y_test_pred = model.predict(X_test_data)


y_pred_labels = le.inverse_transform(y_test_pred) # Inverse predict labels back to adjective words
```

13. 輸出結果

- 保存為提交格式的 CSV 文件。

```
submission = pd.DataFrame({
    'id': test_data['tweet_id'],
    'emotion': y_pred_labels
})


submission.to_csv('C:/Users/vivian/OneDrive - NTHU/文件/清大/Data mining/DMLab2/hw/submission.csv', index=False)
```