Lab2 Report

∨ Import data

```
import json
import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
from collections import Counter
from \quad sklearn. \ feature\_extraction. \ text \quad import \quad TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from \quad sklearn.\,ensemble \quad import \quad Random Forest Classifier
from sklearn.metrics import classification_report
data = []
with open (' \underline{/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()
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')
```

Use data from json to built a dataframe, and sort it by identification

```
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['thashtags']),
        'text': _source.apply(lambda x: x['text']),
})
df = df.merge(data_identification, on='tweet_id', how='left')
train_data = df[df['identification'] == 'train']
test_data = df[df['identification'] == 'test']
train_data = train_data.merge(emotion, on='tweet_id', how='left')
train_data.drop_duplicates(subset=['text'], keep=False, inplace=True)
```

Sampling

I choose 30% random data from the training set cause it is the largest sample the memory can run.

```
train_data_sample = train_data.sample(frac=0.3, random_state=42)

y_train_data = train_data_sample['emotion']

X_train_data = train_data_sample.drop(['tweet_id', 'emotion', 'identification', 'hashtags'], axis=1)
train_data_sample
```

 $\overline{\Rightarrow}$

		tweet_id	hashtags	text	identification	emotion
86	1298	0x29573d	0	@SuicideGirls I will feel high anywhere with t	train	joy
116	2037	0x33a0d6	[Unconscionable]	@SenBobCorker I am a Democrat & I applaud your	train	disgust
12	7464	0x22f649	[ReignCane]	Tulsa showing me some love! <lh> #ReignCane @C</lh>	train	trust
121	6259	0x207fd5	0	@CNN Why now? Why not when obama was president	train	surprise
63	0563	0x34d404		@jtonerv @rajivj @a2zUserGroup Is that a cardb	train	joy
121	6605	0x2fecef	[teamangie]	Finding it diddicult to watch Kerbers matches	train	sadness
24	1188	0x20f525	[Life]	73 The moments in your life are only once #Lif	train	anticipation
27	3638	0x1f9535		@P5HBrazil @FifthHarmony B I vote for @FifthHa	train	sadness
48	8366	0x3707ec	[texas, nyc]	Hate will never win. <lh> #texas #nyc</lh>	train	joy
128	31567	0x2cfb63		Today's travel playlist includes @zoella 's "S	train	joy
434755 rows × 5 columns						

Processing test-train data

Split the test-train set

```
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)
```

Use TF-IDF Vectorizer to transform the data. This place I choose $\max_{\text{features}} = 2000$. The reason is also the memory, 2000 is the biggest number that can run in the memory, or it may crash.

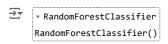
```
tfidf = TfidfVectorizer(max_features=2000)
X = tfidf.fit_transform(X_train['text']).toarray()
X_test = tfidf.transform(X_test['text'])
```

Encode the emootion label, make sure the label can be calculated.

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

Random Forest Classifier

```
classifier = RandomForestClassifier()
classifier.fit(X, y)
```



```
y pred = classifier.predict(X test)
```

Compare the test and the test prediction. Preliminarily watch the accuracy after the above steps.

```
y_test_pred = classifier.predict(X_test_data)
y pred labels = le.inverse transform(y test pred)
y_pred_labels
\Longrightarrow array(['anticipation', 'anticipation', 'sadness', ..., 'anticipation', 'joy', 'sadness'], dtype=object)
submission = pd.DataFrame({
         'id': test_data['tweet_id'],
        'emotion': y_pred_labels
})
submission.\ to\_csv\ ('\ /kaggle/working/submission.\ csv', \quad index=False)
xx = pd.read_csv('/kaggle/working/submission.csv')
XX
₹
                      id
                              emotion
                0x28b412 anticipation
         0
          1
                0x2de201 anticipation
                0x218443
         2
                              sadness
                0x2939d5
          3
                                   joy
          4
                0x26289a
                                  trust
       411967 0x2913b4 anticipation
       411968 0x2a980e anticipation
       411969 0x316b80 anticipation
       411970 0x29d0cb
                                   joy
       411971 0x2a6a4f
                              sadness
      411972 rows × 2 columns
```

Experience of other methods attemping

In the lab, I had also tried BERT, which is the best way to cope with text data I have known, to accomplish the competition. However, there were some problems I had met:

- 1. The memory on kaggle seemed to not enough to run, everytime I run on it the system was crash.
- 2. The training time was excessively long, even when using a smaller batch size or fewer epochs. This made it challenging to iterate and fine-tune the model efficiently.

Hence, I eventually choose the alternative way, Random Forest Classifier, to solve the problem, but I have thought some probable attemps to improve. For example, use lighter version of BERT, or simplify the procedure data processing. Though I haven't succeeded on this competition, I can so that better by this experience.

開始使用 AI 編寫或生成程式碼。