資料探勘研究與實務 HW3 - Sentiment Analysis 1

資管碩一 0853412 吳宛儒

1. 資料前處理

先載入需要的套件：

```
import pandas as pd
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import AdaBoostClassifier
import xgboost as xgb
from sklearn.metrics import classification_report
```

a. 讀取"training_label.txt"與"testing_label.txt"並利用分割符號切割字串、建立 train&test 之 DataFrame

b. 去除停頓詞 stop words

使用套件：

```
import pandas as pd
```

定義 load data 的 function：

```
#%%
def load_data(dataset):
    if dataset == 'train':
        data = pd.read_csv('training_label.txt',sep='\t',header=None)
    elif dataset == 'test':
        data = pd.read_csv('testing_label.txt',sep='\t',header=None)
    data.columns = ['text']
    data['label'] = data['text'].str[0]
    data['data'] = data['text'].str[10:]
    del data['text']
    return data
```

開始 load data，定義 train 以及 test set，train set 只取 train data 前 10000 筆。透過 split_data function 將 data 跟 label 切開來。

```
#%%
# load data
train = load_data('train')
train = train[:][:10000]
test = load_data('test')

def split_data(dataset):
    X, y = [], []
    X = dataset['data'].values
    y = dataset['label'].values
    return X, y

trainX, trainy, testX, testy = [], [], [], []
trainX, trainy = split_data(train)
testX, testy = split_data(test)
```

接著定義 stop words，考量到自己定義 stop words 的 list 有點不切實際，使用到了 nltk 對於 english 的 stop words 定義，將 stop words 設定為 english 的 stop words，並添加其他符號。

註解：這邊 stop word 可以預先定義(如下圖)，也可以在後面的 CountVectorizer 或 TfidfVectorizer 中定義(都會寫出來，但最後跑程式的時候是寫在 Tfidf 中)

```
from nltk.corpus import stopwords
```

```
stop_words = set(stopwords.words('english'))

for w in ['!',',','.','?','-s','-ly','</s>','s']:
    stop_words.add(w)
```

CountVectorizer

原本有使用 CountVectorizer，但是後來在網路上查到其功能 TfidfVectorizer 之 transform 已有包含在內，因此這部分的程式碼在最後並沒有使用。

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
#%%
# 這個部分 tf-idf 的 transform 有包含，所以可以不用做這個 block
cv = CountVectorizer(stop_words='english')
X = cv.fit_transform(trainX)
cool_sw = X.toarray()

feature_sw = cv.get_feature_names()
```

c. 文字探勘前處理，將文字轉換成向量，像是常見的方法 tf-idf、word2vec...等

使用套件：TfidfVectorizer from sklearn

這邊選擇使用 tf-idf 來將文字轉成向量，要設定 stop words 為 english 的 stop words。

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# tf-idf
vectorizer = TfidfVectorizer(stop_words='english')
ldf_train = vectorizer.fit_transform(trainX)
ldf_test=vectorizer.transform(testX)
tfidf_feature = vectorizer.get_feature_names()
print(ldf_train.shape)
```

2. 建模：

AdaBoost

使用套件：AdaBoostClassifier from sklearn

```
from sklearn.ensemble import AdaBoostClassifier
```

建立 classifier，將剛才 tf-idf 做出來的資料拿進去 classifier 做 fit，最後 predict test data 的 label 並印出。

```
# ada
ada = AdaBoostClassifier(n_estimators=100, random_state=0)
ada.fit(ldf_train, trainy)
pred_ada = ada.predict(ldf_test)
```

xgboost

先安裝 XGBoost。

```
import xgboost as xgb
```

建立 XGBoost Classifier，接著用 tf-idf 結果 fit，接著再 predict。

```
# xgbc
xgbc = xgb.XGBClassifier()
xgbc.fit(ldf_train,trainy)
xgbc.score(ldf_test, testy)
pred_xgbc = xgbc.predict(ldf_test)
```

3. 評估模型

利用"testing_label.txt"的資料對 2.所建立的模型進行測試，並計算 Accuracy、Precision、Recall、F-measure

使用套件：sklearn.metrics 中的 classification_report

```
from sklearn.metrics import classification_report
```

```
print('Adaboost:',classification_report(testy, pred_ada))
print('Adaboost:',classification_report(testy, pred_xgbc))
```

AdaBoost 結果：

```
Adaboost:               precision    recall  f1-score   support

           0       0.70      0.43      0.53        37
           1       0.69      0.87      0.77        53

    accuracy                           0.69        90
   macro avg       0.69      0.65      0.65        90
weighted avg       0.69      0.69      0.67        90
```

XGBoost 結果：

```
XGboost:               precision    recall  f1-score   support

           0       0.45      0.24      0.32        37
           1       0.60      0.79      0.68        53

    accuracy                           0.57        90
   macro avg       0.53      0.52      0.50        90
weighted avg       0.54      0.57      0.53        90
```