Text Classification Python

實作報告

流程

- Importing Libraries & Data Set
- Text Preprocessing
- POS & Lemmatisation
- Feature Selection
- Model Training
- Training Text Classification Model and Predicting Category
- Evaluating The Model

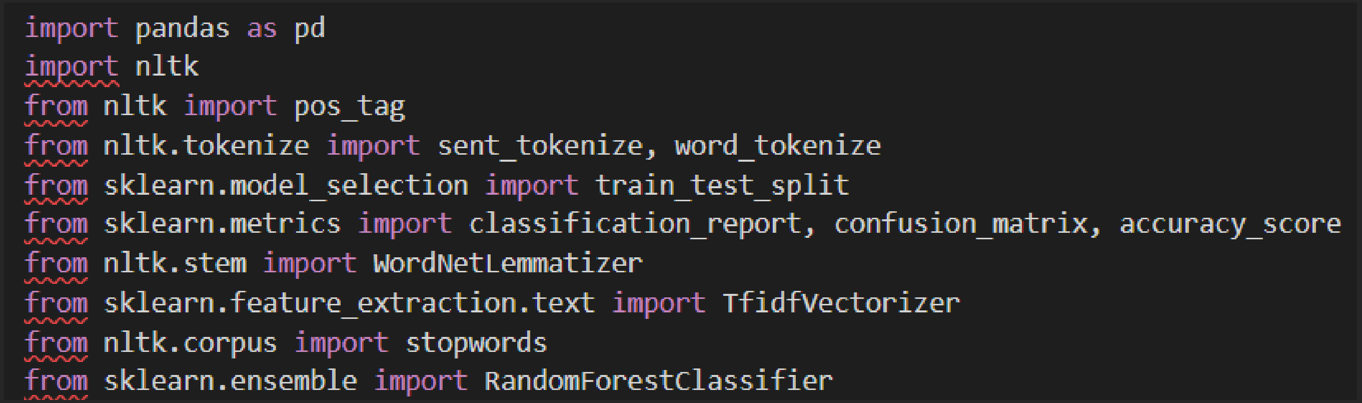
Importing Libraries & Data Set

Mainly use:

Data processing: pandas

Text processing: nltk

ML: sklearn (Random Forest)



```
import pandas as pd
import nltk
from nltk import pos_tag
from nltk.tokenize import sent_tokenize, word_tokenize
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
from sklearn.ensemble import RandomForestClassifier
```

Data Set:

AG collection of news articles

(constructed by Xiang Zhang)

The AG's news topic classification dataset is constructed by choosing 4 largest classes from the original corpus. Each class contains 30,000 training samples and 1,900 testing samples. The total number of training samples is 120,000 and testing 7,600. The classes are: World, Sports, Business and Sci/Tech.

Text Processing

字串處理:

濾掉數字與除了.的標點符號

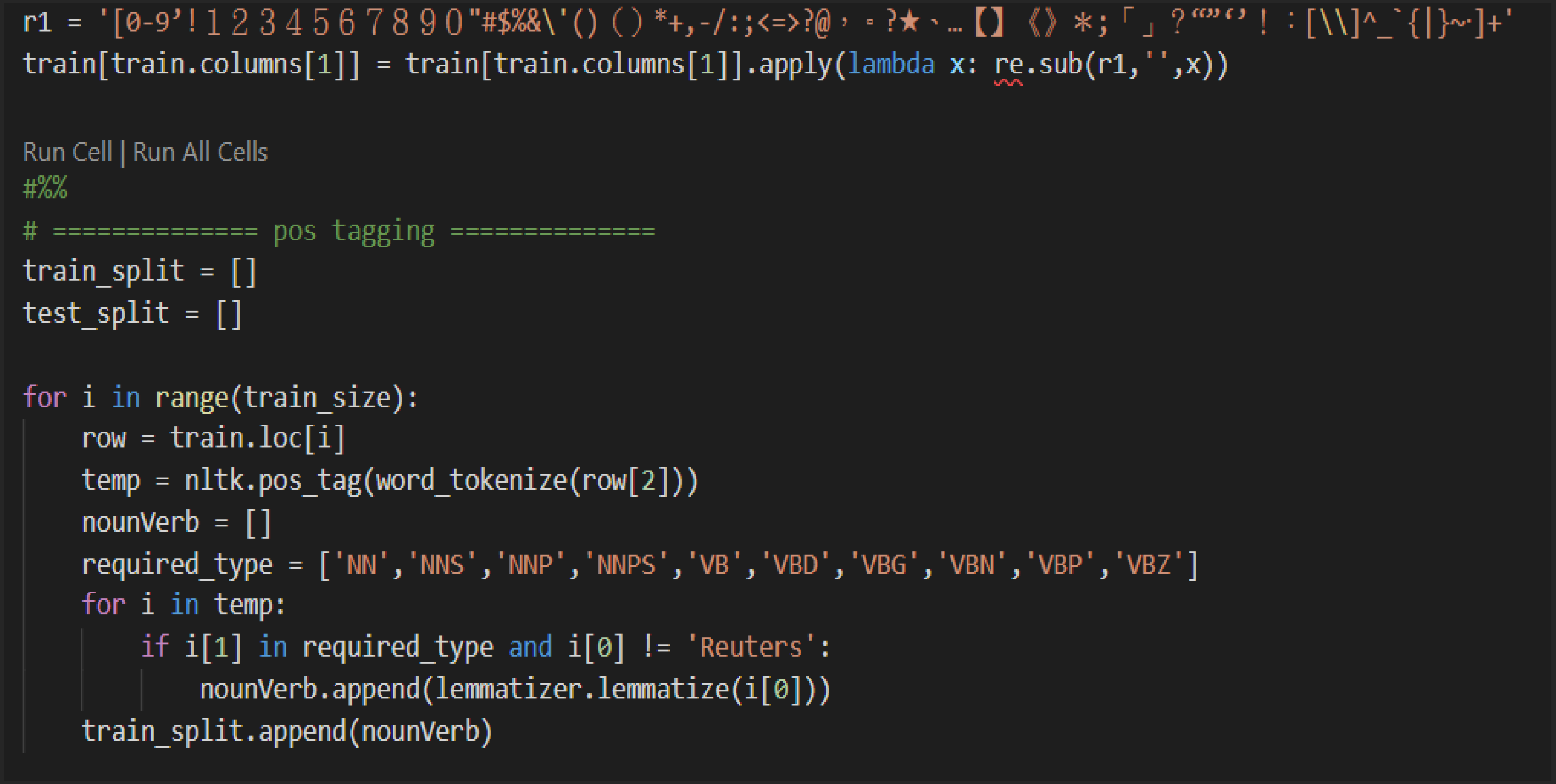
Lemmatisation:

修正文字 - 詞型還原

POS:

只留下我們需要的詞性

（名詞、動詞）

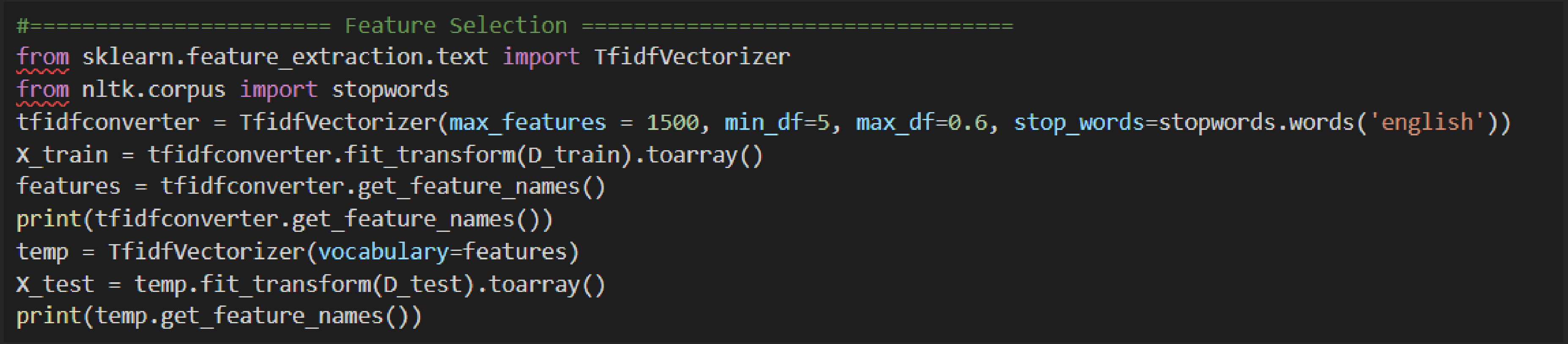


```
r1 = '[0-9’!１２３４５６７８９０"#$%&\'()（）*+,-/:;<=>?@，。?★、…【】《》＊；「」？“”‘’！：[\\]^_`{|}~]+'
train[train.columns[1]] = train[train.columns[1]].apply(lambda x: re.sub(r1,'',x))

Run Cell | Run All Cells
#%%
# ============== pos tagging ==============
train_split = []
test_split = []

for i in range(train_size):
    row = train.loc[i]
    temp = nltk.pos_tag(word_tokenize(row[2]))
    nounVerb = []
    required_type = ['NN','NNS','NNP','NNPS','VB','VBD','VBG','VBN','VBP','VBZ']
    for i in temp:
        if i[1] in required_type and i[0] != 'Reuters':
            nounVerb.append(lemmatizer.lemmatize(i[0]))
    train_split.append(nounVerb)
```

Feature Selection



```
#======================= Feature Selection =================================
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
tfidfconverter = TfidfVectorizer(max_features = 1500, min_df=5, max_df=0.6, stop_words=stopwords.words('english'))
X_train = tfidfconverter.fit_transform(D_train).toarray()
features = tfidfconverter.get_feature_names()
print(tfidfconverter.get_feature_names())
temp = TfidfVectorizer(vocabulary=features)
X_test = temp.fit_transform(D_test).toarray()
print(temp.get_feature_names())
```

挑選Features:

我們選擇使用 TF-IDF 當作選擇Feature時的指標，濾掉詞頻出現過多(超過60%文章出現過)或詞頻過少的單字(80000篇中低於5篇)，挑選出1500個最具代表性的單字作為Feature

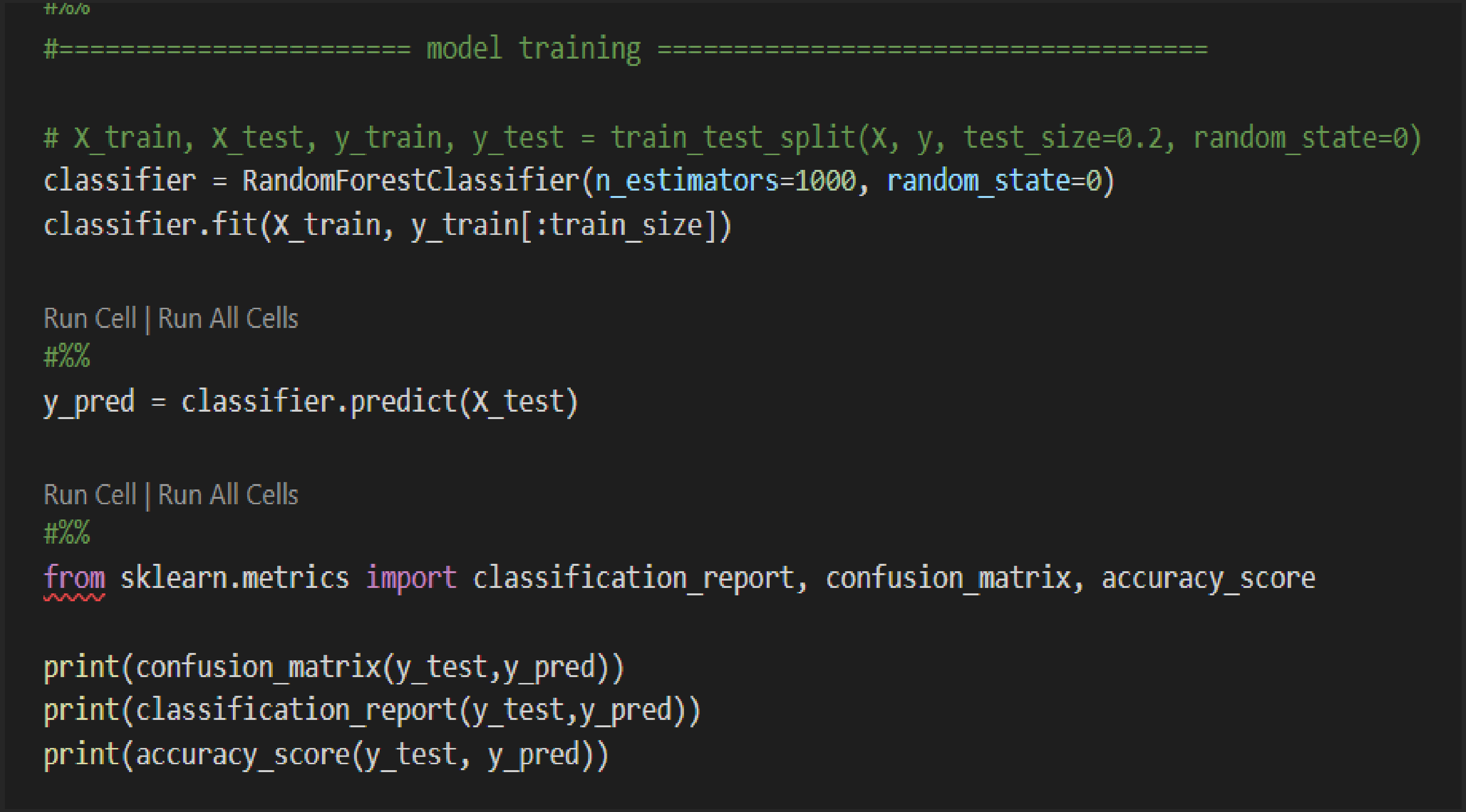
補充:

- 經由tfidfconverter，每篇文章會變成vector形式
- 1500個feature會完全出自於train data 而不是 test data

Model Training & Predicting Category

Random Forest:

用1000個estimators建立模型去預測test data的分類結果



```
#====================== model training ==================================

# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
classifier.fit(X_train, y_train[:train_size])

Run Cell | Run All Cells
#%%
y_pred = classifier.predict(X_test)

Run Cell | Run All Cells
#%%
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

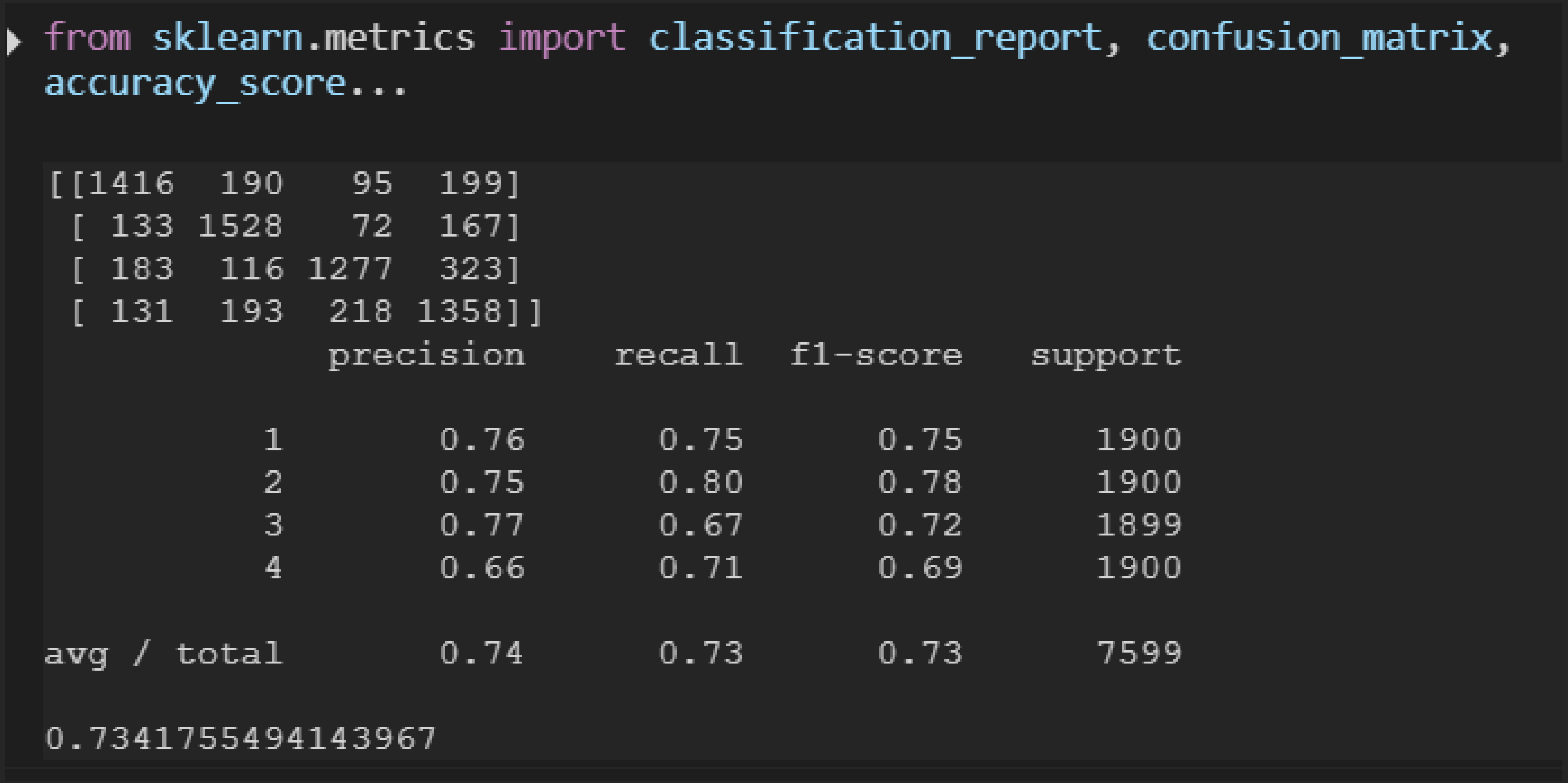
Output (150 features)

Selected features:

```
['according', 'afp', 'agreed', 'announced', 'ap', 'aspx', 'athens', 'attack', 'baghdad', 'bank', 'bush', 'business',
'china', 'city', 'co', 'com', 'company', 'computer', 'consumer', 'corp', 'cost', 'country', 'court', 'cup', 'cut', 'day',
'deal', 'election', 'end', 'executive', 'expected', 'face', 'firm', 'force', 'friday', 'fullquote', 'game', 'get',
'government', 'group', 'gt', 'ha', 'help', 'hit', 'home', 'href', 'http', 'inc', 'including', 'industry', 'international',
'internet', 'investor', 'iraq', 'iraqi', 'job', 'john', 'killed', 'lead', 'leader', 'league', 'london', 'loss', 'lt', 'made',
'make', 'maker', 'market', 'microsoft', 'minister', 'monday', 'month', 'music', 'network', 'new', 'news', 'night',
'official', 'oil', 'olympic', 'open', 'part', 'people', 'percent', 'phone', 'plan', 'player', 'police', 'president', 'price',
'prime', 'product', 'profit', 'quarter', 'quickinfo', 'quot', 'record', 'red', 'report', 'reported', 'research', 'reuters',
'rose', 'run', 'said', 'sale', 'san', 'saturday', 'say', 'search', 'season', 'security', 'series', 'service', 'set', 'share',
'software', 'space', 'start', 'state', 'states', 'stock', 'stocks', 'sunday', 'system', 'take', 'talk', 'target', 'team',
'technology', 'thursday', 'time', 'today', 'tuesday', 'union', 'united', 'us', 'victory', 'wa', 'washington', 'way',
'web', 'wednesday', 'week', 'win', 'world', 'www', 'year', 'yesterday', 'york']
```

Output (150 features)

Total - accuracy: 0.7341755



```
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score...
```

```
[[1416  190   95  199]
 [ 133 1528   72  167]
 [ 183  116 1277  323]
 [ 131  193  218 1358]]
             precision    recall  f1-score   support

          1       0.76      0.75      0.75      1900
          2       0.75      0.80      0.78      1900
          3       0.77      0.67      0.72      1899
          4       0.66      0.71      0.69      1900

avg / total       0.74      0.73      0.73      7599

0.7341755494143967
```

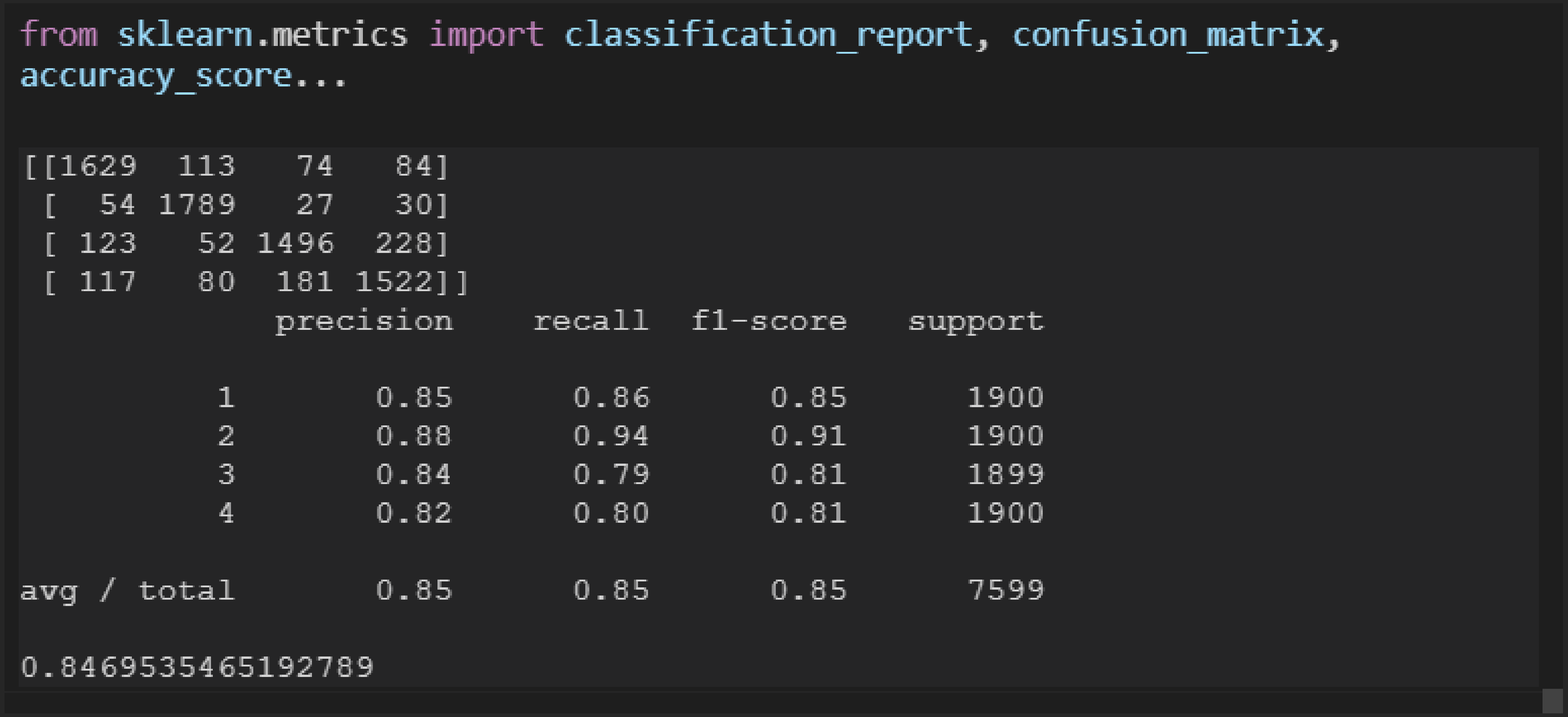
Output (1500 features)

Selected features:

```
['abu', 'access', 'according', 'account', 'accounting', 'accused', 'acquire', 'acquisition', 'act', 'action', 'activity', 'ad',
'add', 'added', 'adding', 'address', 'administration', 'admitted', 'advance', 'advanced', 'advantage', 'advertising',
'afghan', 'afghanistan', 'afp', 'africa', 'african', 'afternoon', 'ag', 'age', 'agency', 'agent', 'agreed', 'agreement', 'aid',
'aim', 'aimed', 'air', 'aircraft', 'airline', 'airlines', 'airport', 'airways', 'al', 'alan', 'alex', 'allegation', 'alliance', 'allow',
'allowed', 'allows', 'amd', 'america', 'american', 'americans', 'amp', 'anaheim', 'analyst', 'andy', 'angeles', 'angels',
'announce', 'announced', 'announcement', 'ap', 'appeal', 'appeared', 'appears', 'apple', 'application', 'approach',
'approval', 'approved', 'arafat', 'area', 'ariel', 'arizona', 'arm', 'army', 'arrest', 'arrested', 'arrived', 'arsenal', 'asia',
'asked', 'aspx', 'assault', 'asset', 'associated', 'associates', 'association', 'astronaut', 'astros', 'athens', 'atlanta',
'attack', 'attempt', 'attention', 'attorney', 'auction', 'aug', 'august', 'australia', 'authority', 'auto', 'average', 'avoid',
'back', 'baghdad', 'ball', 'ballot', 'baltimore', 'ban', 'bank', 'banking', 'bankruptcy', 'barrel', 'barry', 'base', 'baseball',
'based', 'basketball', 'battle', 'bay', 'beat', 'beating', 'became', 'become', 'becoming', 'began', 'begin', 'beginning',
'begun', 'beijing', 'belief', 'believe', 'believed', 'benefit', 'bid', 'big', 'bigley', 'bill', 'bird', 'black', 'blair', 'blast', 'block',
'blood', 'blow', 'blue', 'board', 'body', 'boeing', 'bomb', 'bomber', 'bombing', 'bonds', 'book', 'boost', 'boosted',
'border', 'bos', 'boston', 'bought', 'bowl', 'boy', 'brand', 'braves', 'brazil', 'break', 'bring', 'bringing', 'britain',
'broadband', 'broadcast', 'broke', 'brought', 'brown', 'browser', 'brussels', 'bryant', 'build', 'building', 'built', 'bus',
'bush', 'business', 'buy', 'buying',
```

Output (1500 features)

Total - accuracy: 0.84695354



```
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score...
```

```
[[1629  113   74   84]
 [  54 1789   27   30]
 [ 123   52 1496  228]
 [ 117   80  181 1522]]
             precision    recall  f1-score   support

          1       0.85      0.86      0.85      1900
          2       0.88      0.94      0.91      1900
          3       0.84      0.79      0.81      1899
          4       0.82      0.80      0.81      1900

avg / total       0.85      0.85      0.85      7599

0.8469535465192789
```