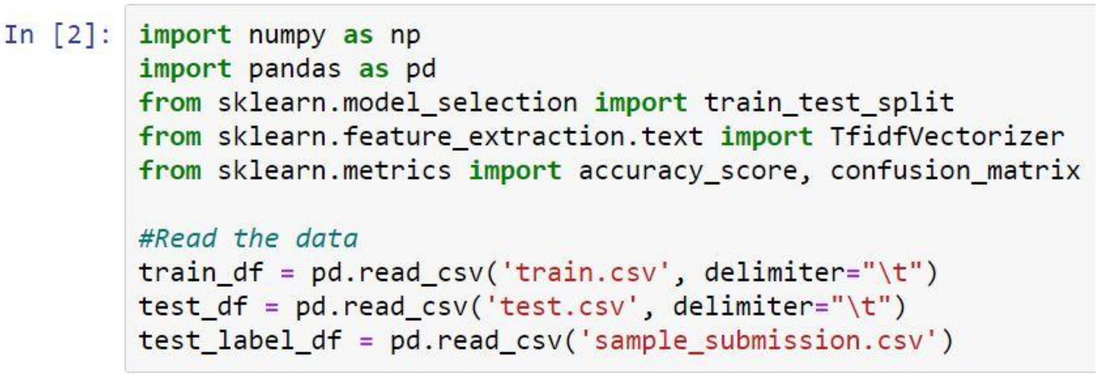
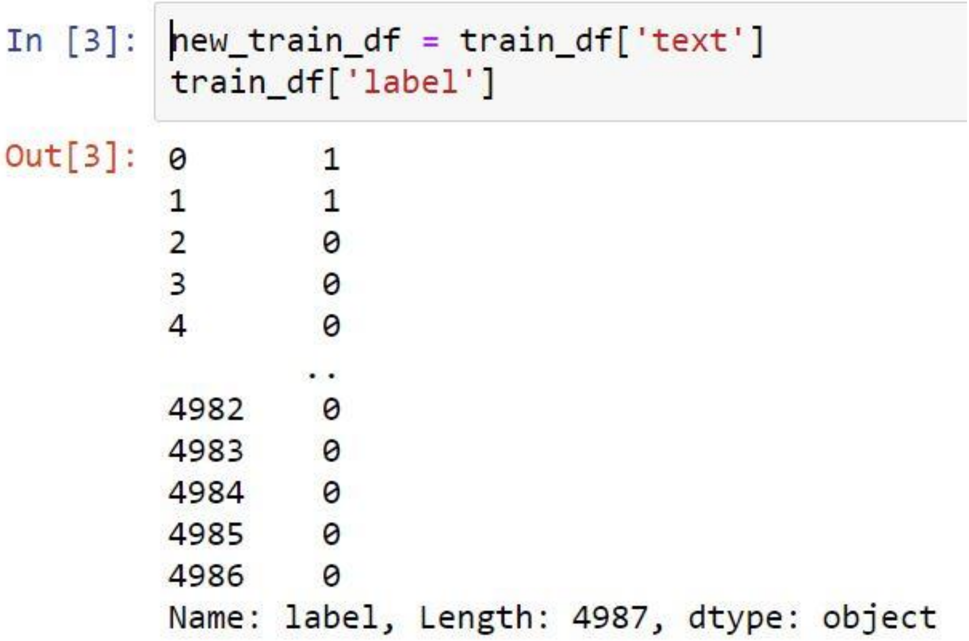


```
In [2]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.metrics import accuracy_score, confusion_matrix

        #Read the data
        train_df = pd.read_csv('train.csv', delimiter="\t")
        test_df = pd.read_csv('test.csv', delimiter="\t")
        test_label_df = pd.read_csv('sample_submission.csv')
```

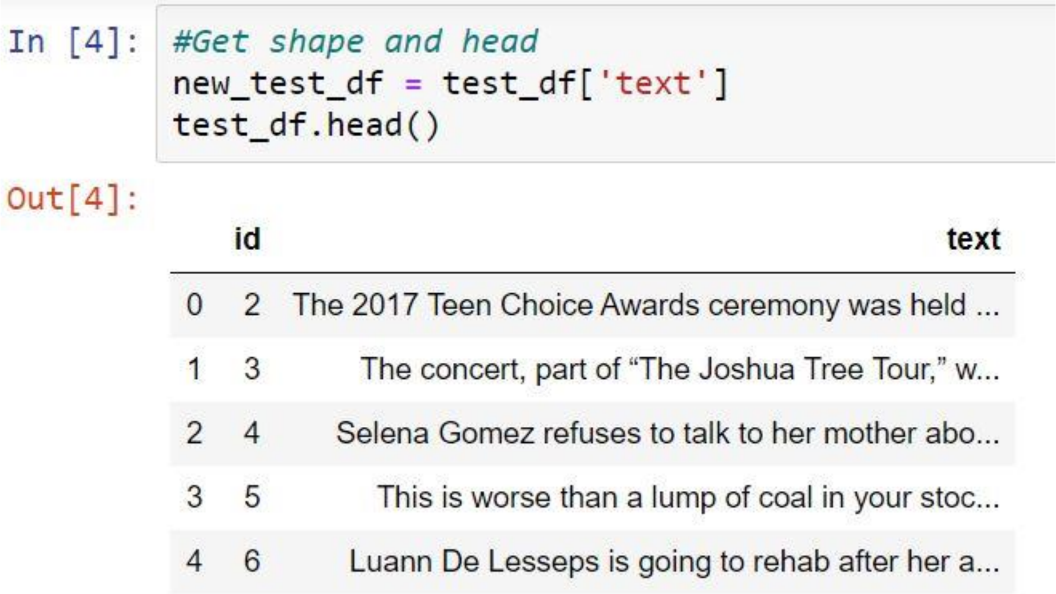
先 import 可能會用到的套件，接著利用 pandas 套件將資料讀取進來，讀取 train.csv、test.csv 和 sample_submission.csv，train.csv、test.csv 利用分割符號切割、建立 train & test 之 DataFrame



```
In [3]: new_train_df = train_df['text']
        train_df['label']

Out[3]: 0       1
        1       1
        2       0
        3       0
        4       0
               ..
        4982    0
        4983    0
        4984    0
        4985    0
        4986    0
        Name: label, Length: 4987, dtype: object
```

因為是進行文字分析，故將 train.csv 的 text 欄位取出放在 new_train_df

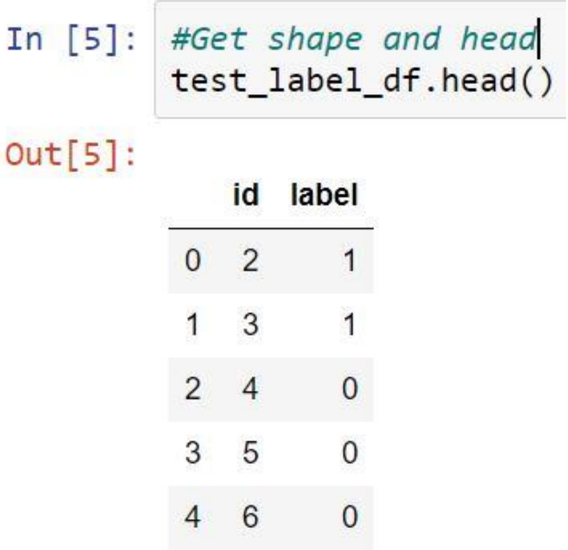


```
In [4]: #Get shape and head
        new_test_df = test_df['text']
        test_df.head()
```

Out[4]:

	id	text
0	2	The 2017 Teen Choice Awards ceremony was held ...
1	3	The concert, part of “The Joshua Tree Tour,” w...
2	4	Selena Gomez refuses to talk to her mother abo...
3	5	This is worse than a lump of coal in your stoc...
4	6	Luann De Lesseps is going to rehab after her a...

將 test.csv 的 text 欄位取出放在 new_test_df

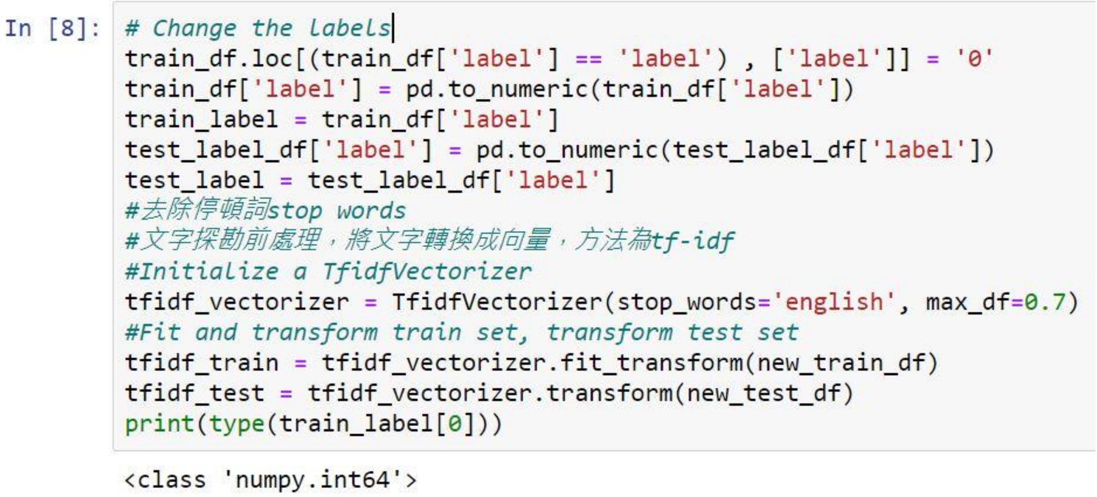


```
In [5]: #Get shape and head
        test_label_df.head()
```

Out[5]:

	id	label
0	2	1
1	3	1
2	4	0
3	5	0
4	6	0

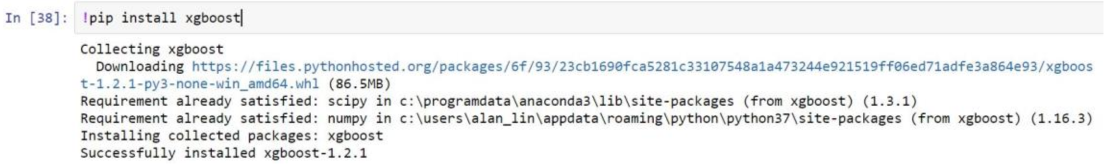
看 sample_submission.csv 的欄位。



```
In [8]: # Change the labels
        train_df.loc[(train_df['label'] == 'label') , ['label']] = '0'
        train_df['label'] = pd.to_numeric(train_df['label'])
        train_label = train_df['label']
        test_label_df['label'] = pd.to_numeric(test_label_df['label'])
        test_label = test_label_df['label']
        #去除停頓詞stop words
        #文字探勘前處理，將文字轉換成向量，方法為tf-idf
        #Initialize a TfidfVectorizer
        tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
        #Fit and transform train set, transform test set
        tfidf_train = tfidf_vectorizer.fit_transform(new_train_df)
        tfidf_test = tfidf_vectorizer.transform(new_test_df)
        print(type(train_label[0]))

        <class 'numpy.int64'>
```

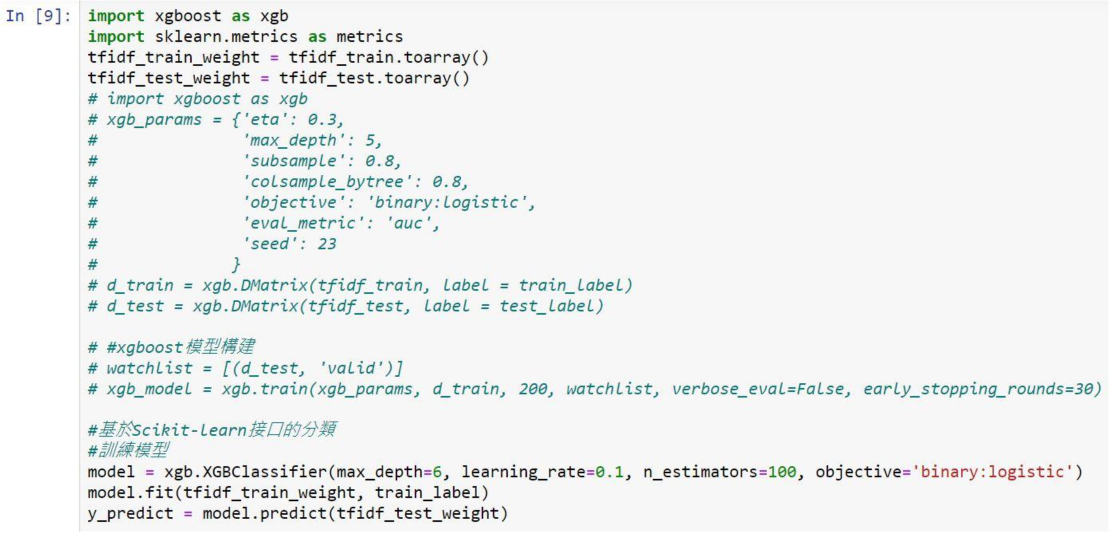
因為 train.csv 的 label 中有一筆資料的值為 label，故我直接將值改為 0，接著將 train_label 和 test_label 的欄位型態轉為 numpy.int64；接著去除停頓詞 stop words，再來進行文字探勘前處理，將文字轉換成向量，我是使用 TfidfVectorizer



```
In [38]: !pip install xgboost

         Collecting xgboost
           Downloading https://files.pythonhosted.org/packages/6f/93/23cb1690fca5281c33107548a1a473244e921519ff06ed71adfe3a864e93/xgboos
         t-1.2.1-py3-none-win_amd64.whl (86.5MB)
         Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from xgboost) (1.3.1)
         Requirement already satisfied: numpy in c:\users\alan_lin\appdata\roaming\python\python37\site-packages (from xgboost) (1.16.3)
         Installing collected packages: xgboost
         Successfully installed xgboost-1.2.1
```

要先 install xgboost 才能 import

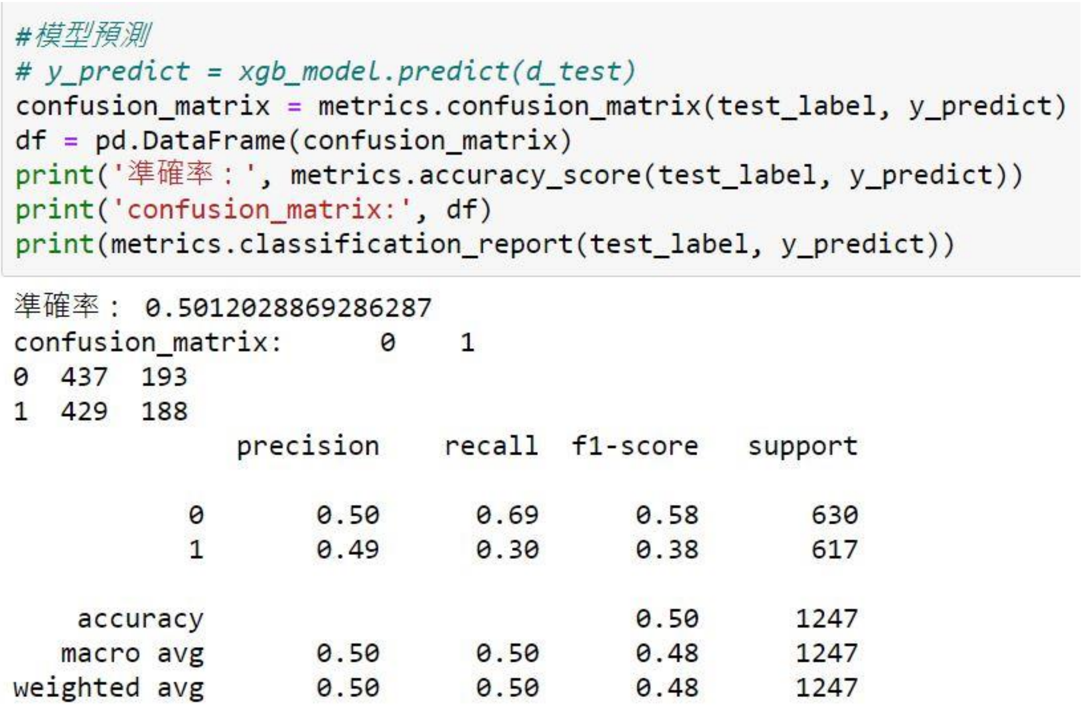


```
In [9]: import xgboost as xgb
        import sklearn.metrics as metrics
        tfidf_train_weight = tfidf_train.toarray()
        tfidf_test_weight = tfidf_test.toarray()
        # import xgboost as xgb
        # xgb_params = {'eta': 0.3,
        #               'max_depth': 5,
        #               'subsample': 0.8,
        #               'colsample_bytree': 0.8,
        #               'objective': 'binary:logistic',
        #               'eval_metric': 'auc',
        #               'seed': 23
        #              }
        # d_train = xgb.DMatrix(tfidf_train, label = train_label)
        # d_test = xgb.DMatrix(tfidf_test, label = test_label)

        # #xgboost模型構建
        # watchlist = [(d_test, 'valid')]
        # xgb_model = xgb.train(xgb_params, d_train, 200, watchlist, verbose_eval=False, early_stopping_rounds=30)

        #基於Scikit-Learn接口的分類
        #訓練模型
        model = xgb.XGBClassifier(max_depth=6, learning_rate=0.1, n_estimators=100, objective='binary:logistic')
        model.fit(tfidf_train_weight, train_label)
        y_predict = model.predict(tfidf_test_weight)
```

使用 xgboost 建模



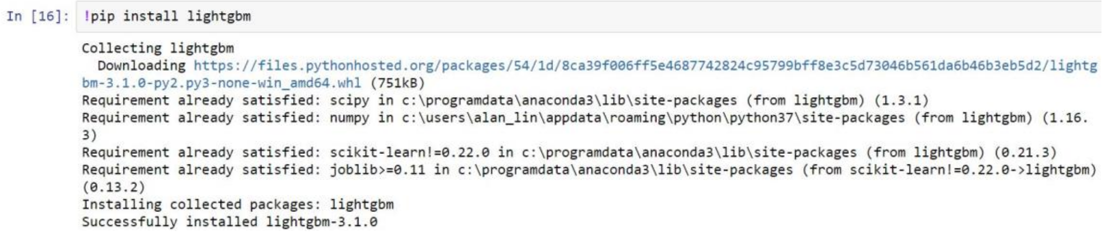
```
#模型預測
# y_predict = xgb_model.predict(d_test)
confusion_matrix = metrics.confusion_matrix(test_label, y_predict)
df = pd.DataFrame(confusion_matrix)
print('準確率：', metrics.accuracy_score(test_label, y_predict))
print('confusion_matrix:', df)
print(metrics.classification_report(test_label, y_predict))
```

```
準確率： 0.5012028869286287
confusion_matrix:      0    1
0  437  193
1  429  188
              precision    recall  f1-score   support

           0       0.50      0.69      0.58       630
           1       0.49      0.30      0.38       617

    accuracy                           0.50      1247
   macro avg       0.50      0.50      0.48      1247
weighted avg       0.50      0.50      0.48      1247
```

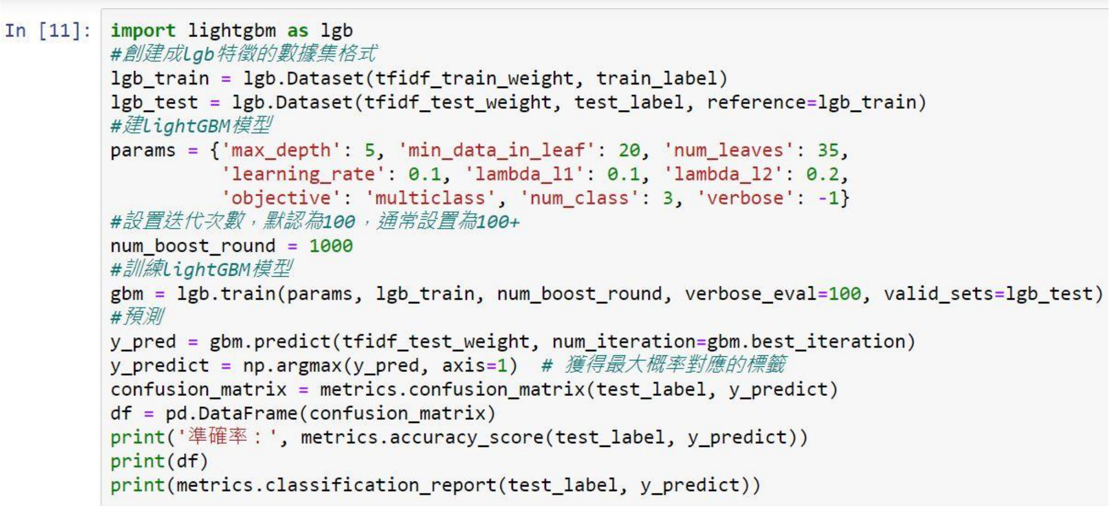
利用"test.csv"的資料對建立的模型進行測試，並計算 Accuracy、Precision、Recall、F-measure



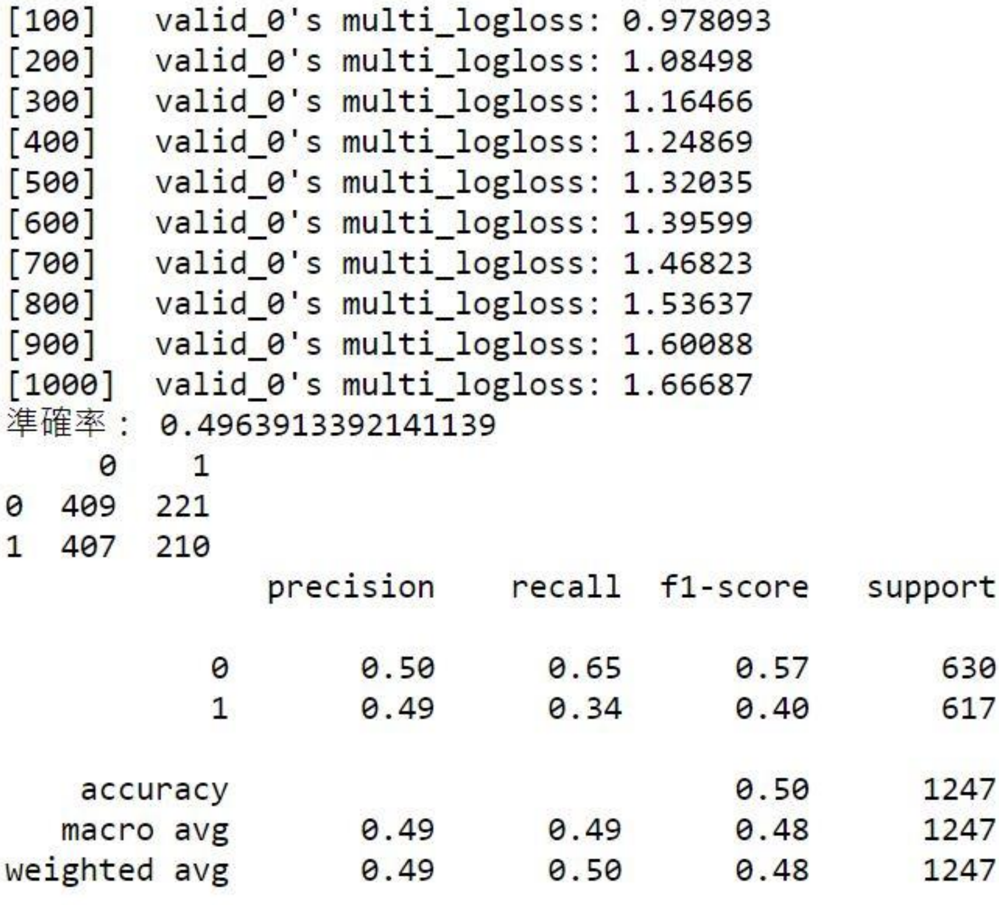
```
In [16]: !pip install lightgbm

Collecting lightgbm
  Downloading https://files.pythonhosted.org/packages/54/1d/8ca39f006ff5e4687742824c95799bff8e3c5d73046b561da6b46b3eb5d2/lightg
bm-3.1.0-py2.py3-none-win_amd64.whl (751kB)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from lightgbm) (1.3.1)
Requirement already satisfied: numpy in c:\users\alan_lin\appdata\roaming\python\python37\site-packages (from lightgbm) (1.16.
3)
Requirement already satisfied: scikit-learn!=0.22.0 in c:\programdata\anaconda3\lib\site-packages (from lightgbm) (0.21.3)
Requirement already satisfied: joblib>=0.11 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn!=0.22.0->lightgbm)
(0.13.2)
Installing collected packages: lightgbm
Successfully installed lightgbm-3.1.0
```

要先 install lightgbm 才能 import



```
In [11]: import lightgbm as lgb
         #創建成lgb特徵的數據集格式
         lgb_train = lgb.Dataset(tfidf_train_weight, train_label)
         lgb_test = lgb.Dataset(tfidf_test_weight, test_label, reference=lgb_train)
         #建lightGBM模型
         params = {'max_depth': 5, 'min_data_in_leaf': 20, 'num_leaves': 35,
                   'learning_rate': 0.1, 'lambda_l1': 0.1, 'lambda_l2': 0.2,
                   'objective': 'multiclass', 'num_class': 3, 'verbose': -1}
         #設置迭代次數，默認為100，通常設置為100+
         num_boost_round = 1000
         #訓練lightGBM模型
         gbm = lgb.train(params, lgb_train, num_boost_round, verbose_eval=100, valid_sets=lgb_test)
         #預測
         y_pred = gbm.predict(tfidf_test_weight, num_iteration=gbm.best_iteration)
         y_predict = np.argmax(y_pred, axis=1)  # 獲得最大概率對應的標籤
         confusion_matrix = metrics.confusion_matrix(test_label, y_predict)
         df = pd.DataFrame(confusion_matrix)
         print('準確率：', metrics.accuracy_score(test_label, y_predict))
         print(df)
         print(metrics.classification_report(test_label, y_predict))
```

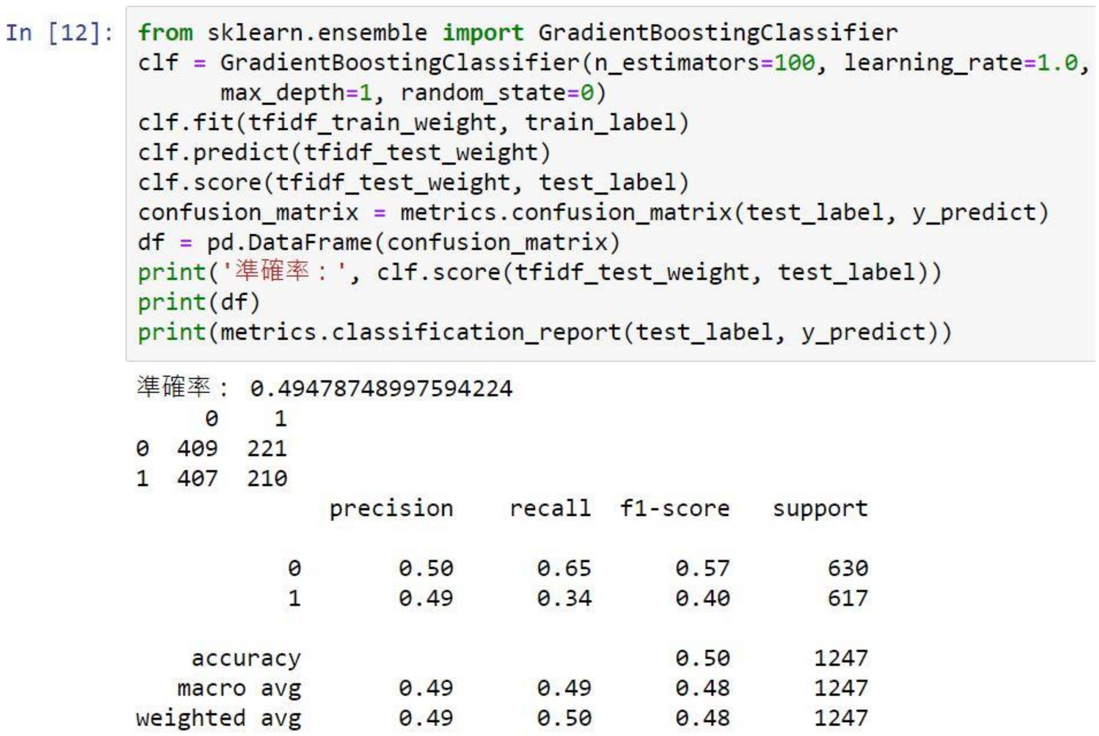


```
[100]   valid_0's multi_logloss: 0.978093
[200]   valid_0's multi_logloss: 1.08498
[300]   valid_0's multi_logloss: 1.16466
[400]   valid_0's multi_logloss: 1.24869
[500]   valid_0's multi_logloss: 1.32035
[600]   valid_0's multi_logloss: 1.39599
[700]   valid_0's multi_logloss: 1.46823
[800]   valid_0's multi_logloss: 1.53637
[900]   valid_0's multi_logloss: 1.60088
[1000]  valid_0's multi_logloss: 1.66687
準確率： 0.4963913392141139
     0    1
0  409  221
1  407  210
              precision    recall  f1-score   support

           0       0.50      0.65      0.57       630
           1       0.49      0.34      0.40       617

    accuracy                           0.50      1247
   macro avg       0.49      0.49      0.48      1247
weighted avg       0.49      0.50      0.48      1247
```

使用 lightgbm 建模，利用"test.csv"的資料對建立的模型進行測試，並計算 Accuracy、Precision、Recall、F-measure



```
In [12]: from sklearn.ensemble import GradientBoostingClassifier
         clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,
               max_depth=1, random_state=0)
         clf.fit(tfidf_train_weight, train_label)
         clf.predict(tfidf_test_weight)
         clf.score(tfidf_test_weight, test_label)
         confusion_matrix = metrics.confusion_matrix(test_label, y_predict)
         df = pd.DataFrame(confusion_matrix)
         print('準確率：', clf.score(tfidf_test_weight, test_label))
         print(df)
         print(metrics.classification_report(test_label, y_predict))
```

```
準確率： 0.49478748997594224
     0    1
0  409  221
1  407  210
              precision    recall  f1-score   support

           0       0.50      0.65      0.57       630
           1       0.49      0.34      0.40       617

    accuracy                           0.50      1247
   macro avg       0.49      0.49      0.48      1247
weighted avg       0.49      0.50      0.48      1247
```

使用 GBDT 建模，利用"test.csv"的資料對建立的模型進行測試，並計算 Accuracy、Precision、Recall、F-measure

GBDT、LightGBM、xgboost 模型之結果比較

GBDT 以 CART 作為基分類器，xgboost 還支持線性分類器，傳統 GBDT 在優化時用到一階導數，xgboost 則對函數進行了二階泰勒展開，在結果表現 xgboost 的準確率會優於 GBDT；LightGBM 的設計就是提供快速高效、低內存占用、高準確度、支持並行和大規模數據處理，再加上 LightGBM 採用 leaf-wise 分裂方法，能產生比 xgboost 所採用的 level-wise 分裂方法更復雜的樹，能使得模型得到更高準確率，故可看出 LightGBM 的準確率會是三者最高。