

1. 標註執行環境

Google colab

2. 程式語言

Python 3

3. 執行方式：

非原生套件：

-From google.colab import drive:讓 Google Drive 中的文件可以在 google.colab 中使用。

-from sklearn.model_selection import train_test_split: 使用這個函數，可以將數據集劃分為訓練集和測試集，以便在機器學習模型的訓練和評估過程中使用

NB:

-from sklearn.naive_bayes import BernoulliNB: 從 scikit-learn 中導入 Bernoulli Naive Bayes 分類器的方式

SVM:

-from scikit-learn import SVM: SVM 是一種監督式機器學習演算法，用於分類和回歸任務。

-from sklearn import metrics: 該模組提供了各種用於評估機器學習模型性能的函數。

-from sklearn.metrics import precision_recall_curve, precision_recall_fscore_support, auc: 導入與 Precision-Recall Curve 和其他評估指標相關的特定函數。precision_recall_curve 計算不同概率閾值的 precision-recall 配對，precision_recall_fscore_support 計算 precision、recall 和 F1 分數，auc 計算曲線下面積。

-import matplotlib.pyplot as plt: 從 matplotlib 庫中導入 pyplot 模組，用於創建可視化，例如圖表和圖形。

-from sklearn.preprocessing import label_binarize: 該函數用於將分類的類別標籤轉換為二進制向量。

-from sklearn.multiclass import OneVsRestClassifier 這是一種將二元分類算法擴展到多類問題的策略。

-import numpy as np:-

原生套件：

-引入 python 標準庫 os，用於操作檔案和資料夾，方便讀取或寫入檔案。

執行輸出：

-點擊 google colab 中的執行階段，點全部執行或者按 ctrl+F9 便可執行程式

碼，作業要求的 Cosine Similarity 會輸出在 Command Line，TF-IDF vectors 則會存取進雲端中的 TF-IDF-Vectors"。

4. 作業處理邏輯說明：

導入相關模組和設定 Google Colab 掛載：

載入文檔標籤：

```
-classes = [list(map(int, line.split())) for line in file]:
```

最終，classes 是一個包含多個子列表的列表，每個子列表代表一個標籤，第一個元素是類別標籤，其餘的元素是相應的文檔編號。

載入文本數據和標籤：

```
with open(os.path.join(document_folder, f"{i}.txt"), 'r',  
encoding='utf-8') as file:
```

```
    content = file.read()
```

NB: 載入 PAI-data folder

SVM: 載入 TF-IDF-Vectors folder

NB: 使用二元計數向量化文本數據：

```
binary_vectorizer = CountVectorizer(binary=True)
```

使用 scikit-learn 的 `CountVectorizer` 將文本數據轉換為二元計數向量，這表示每個單詞的存在與否

建立訓練和測試集：

y.append(cls): 將對應文檔編號的標籤 (cls) 添加到 y 列表中。

```
x_train, x_test, y_train, y_test = train_test_split(x, y,  
test_size=0.1, stratify=y, random_state=42):
```

train_test_split 函數用於將資料集分為訓練集和測試集。

test_size=0.1 表示將 10% 的數據分配給測試集。

stratify=y 表示根據標籤進行分層劃分，以確保訓練集和測試集中的類別分佈相似。

random_state=42 是為了確保每次運行時切割的結果都是相同的。

建立模型：

```
NB: model = BernoulliNB()
```

```
model.fit(x_train, y_train)
```

```
SVM(Linear): SVM_model = SVC(kernel='linear', C=1.0,  
probability=True)
```

```
SVM(rdf): SVM_model = SVC(kernel='rbf', gamma='scale', C=1.0,  
probability=True)
```

```
SVM_model.fit(x_train, y_train)
```

進行預測和評估模型：

```
predicted_results = model.predict(x_test)
expected_results = y_test
print(metrics.classification_report(expected_results, predicted_results))
```

繪製多類別的精確度-召回曲線：

1) 二值化標籤：

```
y_test_bin = label_binarize(y_test, classes=np.unique(y))
```

使用 `label_binarize` 函數將原始的多類別標籤 (`y_test`) 轉換成二元矩陣。

2) 計算每個類別的精確度-召回曲線：

```
precision, recall, _ = precision_recall_curve(y_test_bin[:, i],
y_score[:, i])
auc_score = auc(recall, precision)
plt.plot(recall, precision, lw=2, label=f'Class {i+1} (AUC = {auc_score:.2f})')
```

使用迴圈遍歷每個類別，計算精確度-召回曲線的數據點，並計算曲線下面積 (AUC)。每條曲線都用不同的標籤和線寬繪製。

3) 圖形設定：

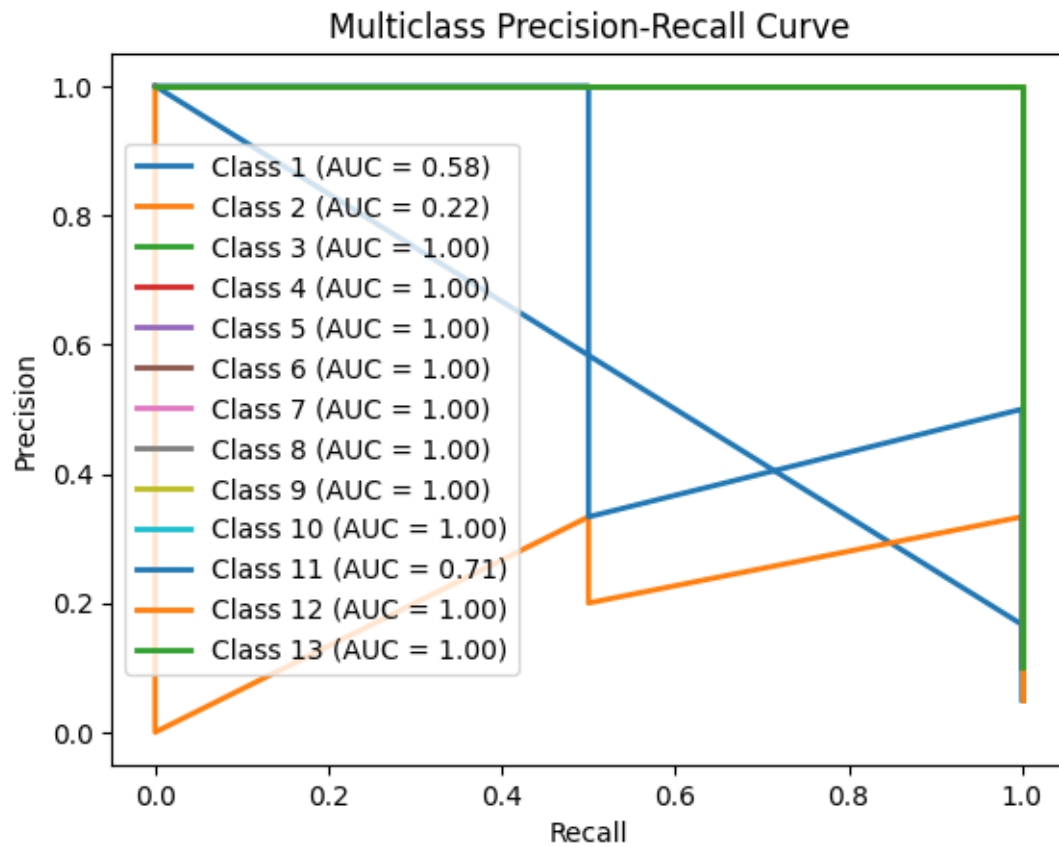
設定 x 軸為召回率，y 軸為精確度。添加標題、軸標籤和圖例，最後使用 `plt.show()` 顯示圖形。

5. 結果：

NB:

	precision	recall	f1-score	support
1	0.17	1.00	0.29	1
2	0.00	0.00	0.00	2
3	1.00	1.00	1.00	1
4	0.00	0.00	0.00	2
5	1.00	1.00	1.00	2
6	1.00	1.00	1.00	1
7	1.00	1.00	1.00	2
8	1.00	1.00	1.00	2
9	1.00	1.00	1.00	1
10	1.00	1.00	1.00	1
11	1.00	0.50	0.67	2
12	0.50	1.00	0.67	1

	13	1.00	0.50	0.67	2
accuracy				0.70	20
macro avg		0.74	0.77	0.71	20
weighted avg		0.73	0.70	0.68	20

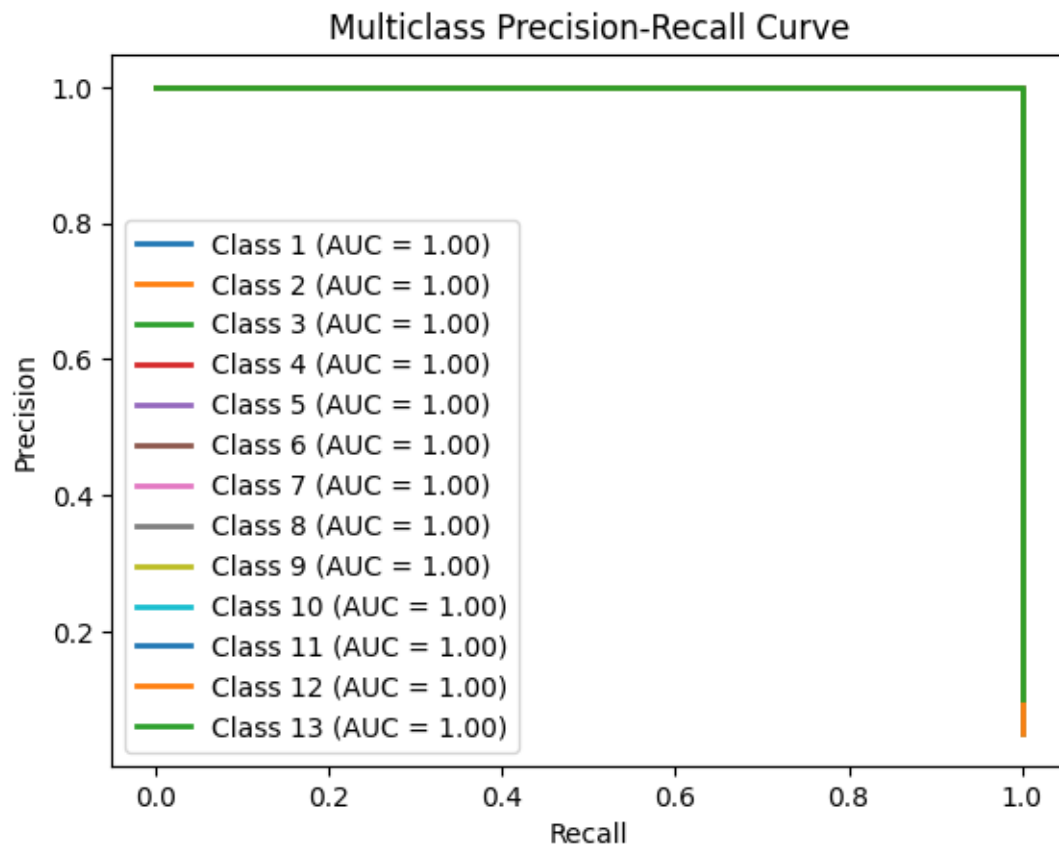


SVM(Leaner)

precision recall f1-score support

1	1.00	1.00	1.00	1
2	1.00	1.00	1.00	2
3	1.00	1.00	1.00	1
4	1.00	1.00	1.00	2
5	1.00	1.00	1.00	2
6	1.00	1.00	1.00	1
7	1.00	1.00	1.00	2
8	1.00	1.00	1.00	2
9	1.00	1.00	1.00	1
10	1.00	1.00	1.00	1

11	1.00	1.00	1.00	2
12	1.00	1.00	1.00	1
13	1.00	1.00	1.00	2
accuracy			1.00	20
macro avg	1.00	1.00	1.00	20
weighted avg	1.00	1.00	1.00	20

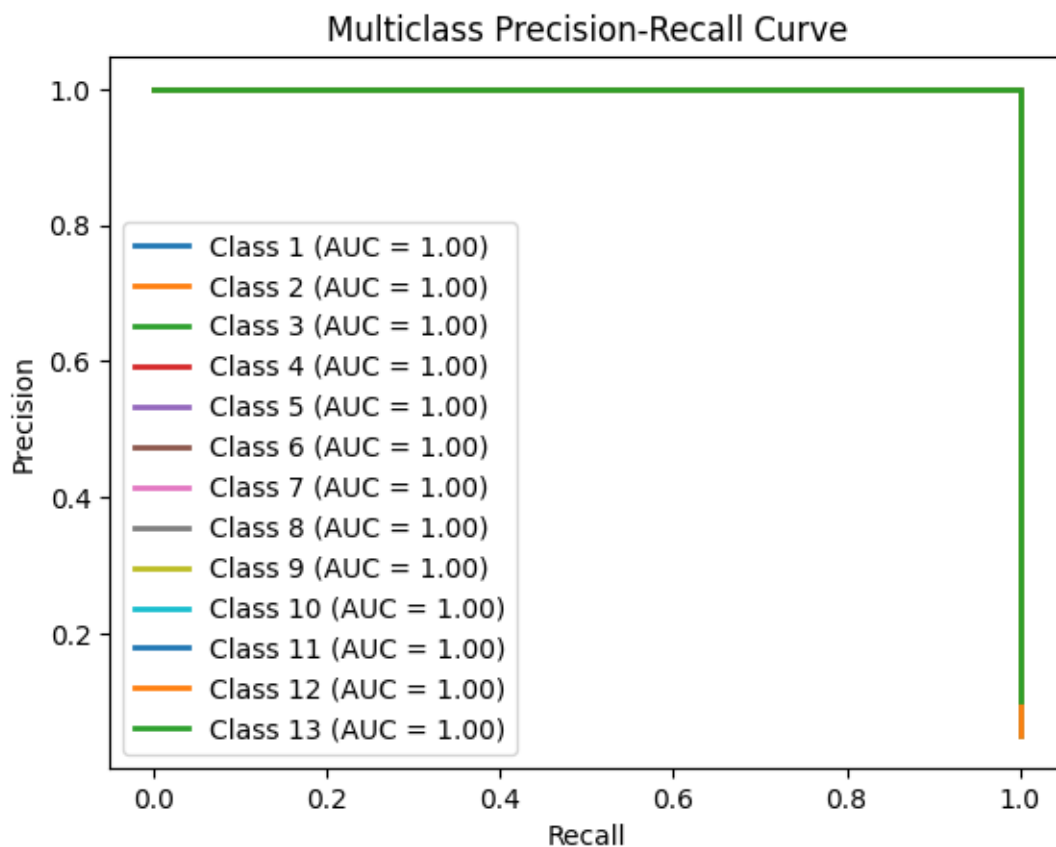


SVM(RBF):

precision recall f1-score support

1	1.00	1.00	1.00	1
2	1.00	1.00	1.00	2
3	1.00	1.00	1.00	1
4	1.00	1.00	1.00	2
5	1.00	1.00	1.00	2
6	1.00	1.00	1.00	1
7	1.00	1.00	1.00	2
8	1.00	1.00	1.00	2
9	1.00	1.00	1.00	1

	10	1.00	1.00	1.00	1
	11	1.00	1.00	1.00	2
	12	1.00	1.00	1.00	1
	13	1.00	1.00	1.00	2
accuracy				1.00	20
macro avg		1.00	1.00	1.00	20
weighted avg		1.00	1.00	1.00	20



精確度 (Precision)：

精確度是指在所有被模型預測為正例的樣本中，實際上是正例的比例。

公式： $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$

其中，TP (True Positive) 是模型正確預測為正例的樣本數，FP (False Positive) 是模型錯誤預測為正例的樣本數。

精確度的值範圍在 0 到 1 之間，越接近 1 表示模型在預測正例方面的效果越好。

召回率 (Recall)：

召回率是指在所有實際為正例的樣本中，模型成功預測為正例的比例。

公式： $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$

其中，TP (True Positive) 是模型正確預測為正例的樣本數，FN (False Negative) 是模型錯誤預測為負例的樣本數。

召回率的值範圍在 0 到 1 之間，越接近 1 表示模型在捕捉正例方面的效果越好。




AUC：

AUC 是 ROC 曲線下的面積，取值範圍在 0 到 1 之間。AUC 越接近 1，表示模型性能越好。

AUC 的解釋是在隨機選取一個正例和一個負例的情況下，模型正確區分它們的概率。AUC 等於 0.5 表示模型的預測效果等同於隨機猜測，大於 0.5 表示模型優於隨機猜測，越接近 1 表示模型性能越好。

由上評估：

使用 SVM 模組的預測效果更好，上傳結果至 kaggle 也可以發現使用 SVM 模組分數以及精確率會高於使用 NB。

	output2.csv Complete · 3d ago	0.97777	<input type="checkbox"/>
	output1.csv Complete · 3d ago	0.93611	<input type="checkbox"/>
	output (1).csv Complete · 3d ago	0.67361	<input type="checkbox"/>

由上至下，分別為使用 SVM(linear), SVM(rbf)以及使用 NB 的預測結果。