書面報告

壹、主題

Spotify 讀書歌單推薦系統

貳、題目發想及動機

許多同學在學習或閱讀時都喜歡聆聽音樂，因為音樂能夠提供舒適的背景與適度的娛樂性，幫助專注、提升效率以達成良好的學習狀態。然而，要找到合適的歌單，或是自行建立歌單：邊聽邊篩選歌曲再一首一首加入播放清單中，相當費時。

Spotify 是全球擁有最多使用者的音樂串流平台，擁有龐大的音樂資料庫以及強大的API與活躍的開發者社群，因此我們選定Spotify作為本次報告使用的音樂串流平台。

Spotify 本身已有多元的歌單，其中包含官方及社群分享的 Study Playlist，但官方歌單更新速度極慢，而社群分享的 Study Playlist 品質不一（部分收錄歌曲的曲調、音量、節奏...等，明顯不適合邊讀書邊聽），因此我們將進行資料分析及建立機器學習模型，使用音樂的曲調、音量、節奏、舞蹈性**.....**等 **12** 種變項，為使用者生成推薦的讀書歌單，因此，我們將期末報告的主題訂為「**Spotify** 讀書歌單推薦系統」，預計建立模型並透過前後端整合，以網頁的方式呈現我們的系統。

參、資料蒐集

蒐集流程:

1. 先以 “Study” 作為搜尋關鍵字, 手動整理出Spotify官方推薦的讀書歌單以及社群追蹤數超過10000的讀書歌單(考量官方資料完整性及時下追蹤趨勢, 蒐集到的歌曲以英語、韓語歌曲為主)。
2. 經過小組討論後, 我們整理出可能不適合邊讀書時邊聽的歌曲類型:吵雜、節奏變化頻率高、舞蹈性太高, 並以 “Heavy Metal”, “Rock”, “Party”, “Work out” 作為搜尋關鍵字, 手動整理出Spotify官方推薦以及社群追蹤數超過10000的歌單。
3. 實作爬蟲的過程中, 利用Spotify 所提供的 Audio Features API 來取得每首歌的變項, 如:音高、節奏、音量、能量等等共計 12 種變項。初步蒐集完成後, 得到兩個歌單資料集: “Study Playlist”(適合在學習時聆聽的歌曲)以及 “Non - Study Playlist”(非適合讀書歌曲)。
4. 採取「容許偽陽性、減少偽陰性」的原則整併資料:比對兩個資料集, 若Non - Study Playlist資料集初步蒐集到的歌曲有被收錄在Study Playlist中, 則將其視為適合讀書的歌曲, 從Non - Study Playlist中刪除。

肆、資料清理

由於資料集是根據Spotify上提供的特定格式資料爬取, 因此, 我們預想在多數情況下並不會有資料缺失的問題, 但在我們實作過程進行分析或訓練時, 仍有遇到少數型別上的不相容或是空值。

為了解決型別上的不相容或是空值, 我們利用 Pandas 將不相容的數值型態的欄位整理出來, 將含有空欄位的資料列移除;若出現格式不符的資料, 檢視是否僅是由部份的資料誤植所導致, 若是則手動修正, 而無法處理則逕行刪除。

查閱 Spotify 官方所提供的 Web API Refernece Documentation, 發現音高 (key)、調性 (mode) 以及音符值 (time_signature) 屬於類別變數, 因此使用 get_dummies 以 **one-hot encoding** 的方式取得虛擬變數。最後, 使用 Normalize 將資料中的各變項進行正規化。

伍、資料分析

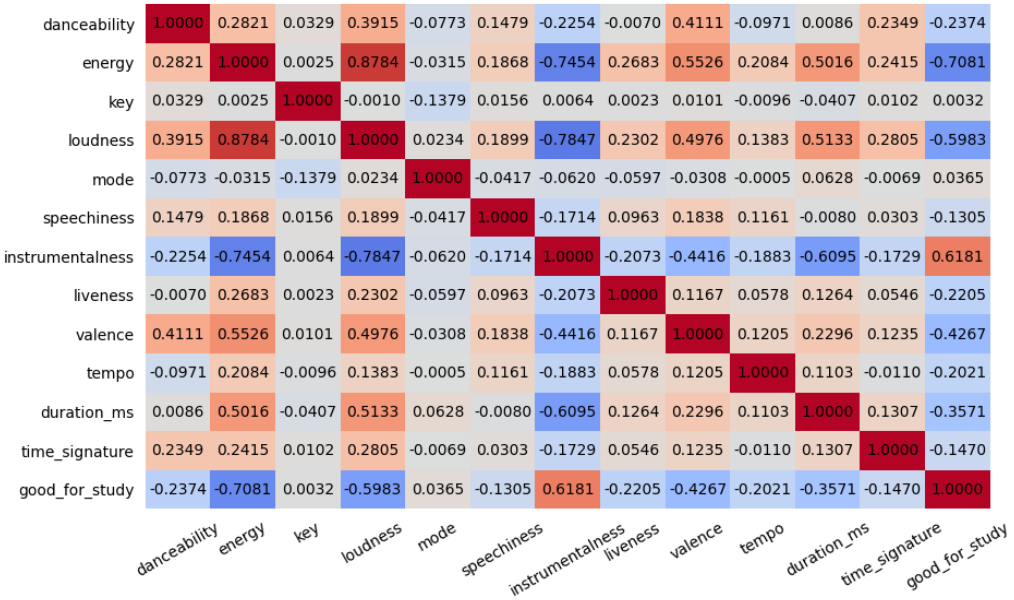
一、資料欄位

欄位名稱	型別	平均值	標準差	最小值	最大值
danceability 舞蹈性	float64	0.626	0.146	0.078	0.961
energy 衝擊性	float64	0.565	0.276	0.005	0.994
key 音高	int64	5.326	3.616	0.000	11.000
loudness 音量(分貝, 基值:-60)	float64	-9.227	6.833	-33.782	-0.005
mode 大調／小調	int64	0.561	0.496	0.000	1.000
speechiness 朗誦佔比	float64	0.084	0.076	0.024	0.611
instrumentalness 純音樂性(無歌詞)	float64	0.303	0.411	0.000	0.981
liveness 臨場感	float64	0.173	0.132	0.022	0.923
valence 正向情緒	float64	0.442	0.236	0.035	0.972
tempo 節拍／每分鐘	float64	119.108	29.856	39.874	209.123
duration_ms 時長(毫秒)	int64	181227.0	45851.4	35375.0	358271.0
time_signature 音符值(相對持續時間)	int64	3.906	0.397	1.000	5.000
good_for_study 適合於學習時聆聽(標記)	int64	0.533	0.499	0.000	1.000

二、特徵相關圖

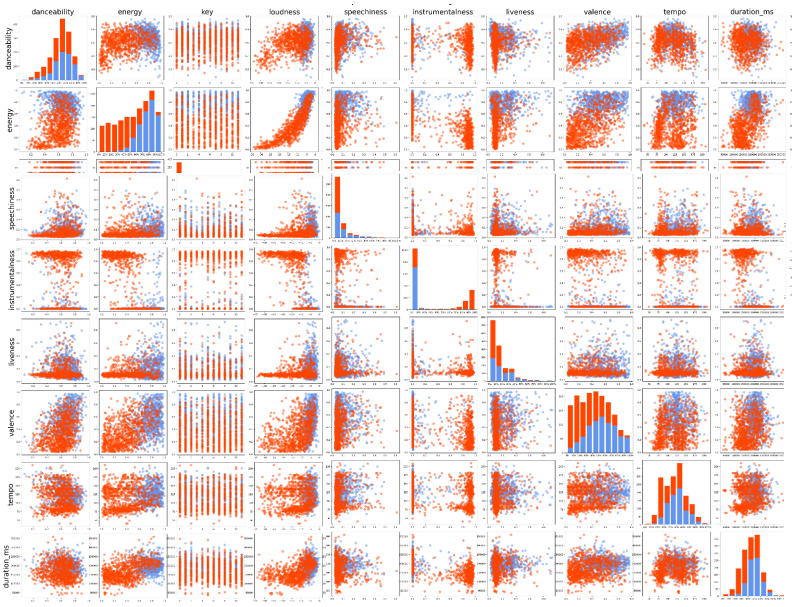
使用相關性矩陣查看資料集中各個特徵的相關程度，其中與 good_for_study 相關性絕對值大於 0.5 的特徵為：energy, loudness, instrumentalness。

如圖：



三、特徵分布圖

雖然由上圖得知特徵的關聯性，卻不知在訓練集中的適合讀書的歌曲與非適合讀書的歌曲的資料特徵分布，因此，在下圖中依照將特徵兩兩分組，用不同顏色標示資料特徵分布：（此為刪除二元數值的變項：mode、time_signature、good_for_study 後所繪製的特徵分布圖版本，完整版請見程式碼）



陸、模型

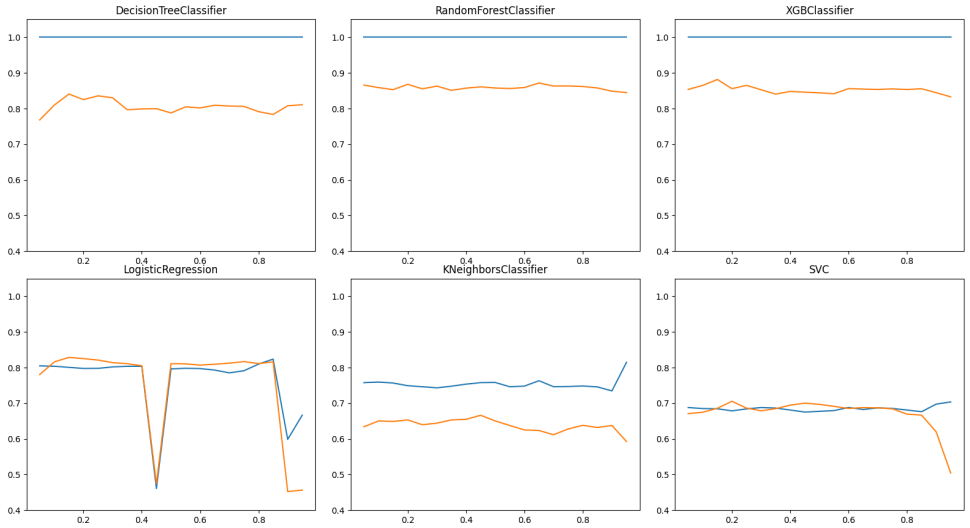
一、資料整併

沒有被收錄在Study Playlist中的歌曲不一定不適合邊讀書邊聽，而是官方或建立熱門歌單者沒有將其選入Study Playlist中，但若有被收錄在Study Playlist中，則代表有人會認為這樣的歌曲適合邊讀書邊聽;另外，在這個推薦歌單的系統中，若將「適合讀書」定為陽性、「不適合讀書」為陰性，則應減少偽陰性，因為使用者在歌單中聽到認為不合適的歌曲時，可直接刪除，故希望生成的歌單中能有愈多可能適合邊讀書邊聽的歌曲愈佳。

因上述原因，採取「容許偽陽性、減少偽陰性」的原則整併資料：比對 Study Playlist 資料集以及 Non - Study Playlist 資料集，若 Non - Study Playlist 資料集初步蒐集到的歌曲有被收錄在 Study Playlist 中，則將其視為適合讀書的歌曲，從 Non - Study Playlist 中刪除。最後，利用 **Shuffle** 將資料隨機排序，以避免分割訓練集與驗證集時，所分割的資料good_for_study值不平均。

二、建模方式：選擇適當模型

在本階段預測目標為「是否適合讀書的音樂」，是二元分類問題，因此使用分類模型：DecisionTreeClassifier、RandomForestClassifier、XGBClassifier、LogisticRegression、KneighborsClassifier 以及 SVC 共計六種機器學習分類方式進行模型表現評估，並繪製根據學習資料比例，所呈現的學習分數。（下圖為使用所有變項的版本，另有僅採取相關性大於 0.2 的變項的版本，請見程式碼。）



從上圖結果觀察出前四者（DecisionTree、RandomForest、XGBoost，LogisticRegression）有較好的表現，經過資料蒐集調查，XGBoost 的 booster 參數可以使用gbtree（梯度提升樹），效果較隨機森林與決策樹具有更高的預測效果以及魯棒性。

因此，小組決定使用 **XGBoost（將 booster 參數設置為 gbtree）** 和 **Logistic-Regression** 來進行下階段的模型評估與選擇。

三、建模與調整參數:

- XGBoost Model

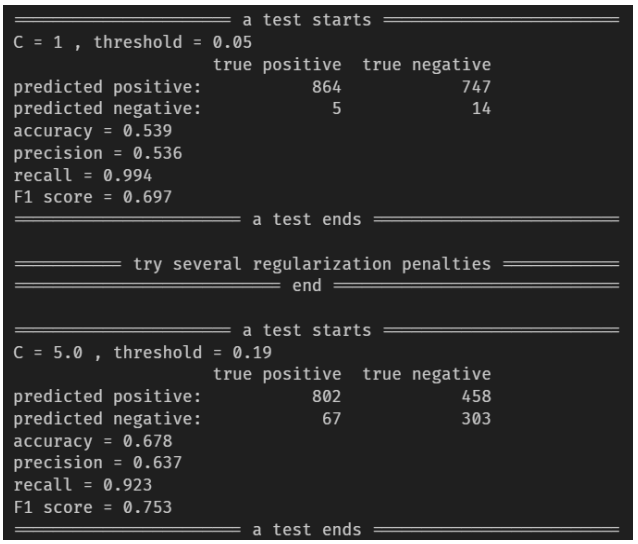
XGBoost 有原生介面以及 Sklearn 介面的分類器方式, 儘管在參數相同情況下, 執行效能應當相同, 但根據不同的介面在算法上會造成些許差異, 模型成果段落將會實作兩者, 並表示其在準確性上的落差及特徵重要性排序的不同。

嘗試進行 Grid 交叉驗證, 但參數過多影響執行效率, 因此也有嘗試隨機方式抽取, 但效果不如手動調參的成效佳, 故最後參數經由手動調參完成。

參數決定考量:**booster** 採用 **gbtree** 分類方式會達到最好的效果、二元分類問題, 因此將 num_class 設為 2;gamma 為避免參數過擬和, 在 { 0.1, 0.2, 0.3 } 分別測試得出 0.1 最佳;max_depth 也同為避免過擬和, 在 { 3, 4, 5, 6 } 得出 6 最佳;sub_sample 設置 0.7 表示每棵樹使用的訓練數據為原始數據集的 70%, 提高模型泛化能力、eta(學習率)分別測試 { 0.01, 0.1, 1 } 得到 0.1 最佳。

- Logistic Regression

先以 C 值為 1、陽性門檻值為 0.05 建立一個模型作為比較基準, 再使用迭代法尋找最佳**C**值陽性門檻值組合, 透過自訂函式建立模型, 並在函式內進行 Cross validation prediction, 然後以格式化輸出呈現混淆矩陣, 計算出 Accuracy、Precision、Recall、F1 score 四個評估分數, 如圖:



雖然在整併資料集的原則是容許容許偽陽性、減少偽陰性, 但在調整參數的過程中若以 **Recall** 值愈高愈好作為調整的原則, 會導致最佳模型將所有資料都預測為陽性, 因此模型評估方式採用以下兩種標準:原始資料比對、**F1 score**。

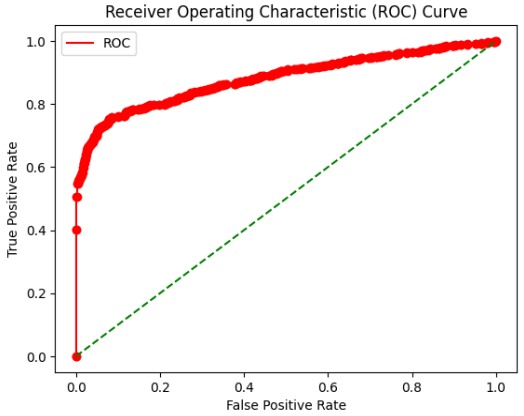
柒、模型的結果及成果

- Logistic Regression

  進行交叉驗證 **(10 folds)**，計算10次驗證所得之 F1-score 及 Accuracy，取其最大值、最小值、平均值以觀察模型表現：

```
========== F1-Score ==========
Cross-validation F1-scores: [0.82840237 0.80722892 0.82926829 0.81528662 0.79532164 0.8
 0.82208589 0.77575758 0.83333333 0.78787879]
Min F1-score:  0.776
Mean F1-score:  0.809
Max F1-score:  0.833
========== Accuracy ==========
Cross-validation accuracy: [0.82208589 0.80368098 0.82822086 0.82208589 0.78527607 0.80368098
 0.82208589 0.77300613 0.82822086 0.78527607]
Min accuracy:  0.773
Mean accuracy:  0.807
Max accuracy:  0.828
```

  使用 ROC 曲線繪製模型在不同閾值下分類性能，評估模型的整體預測能力：



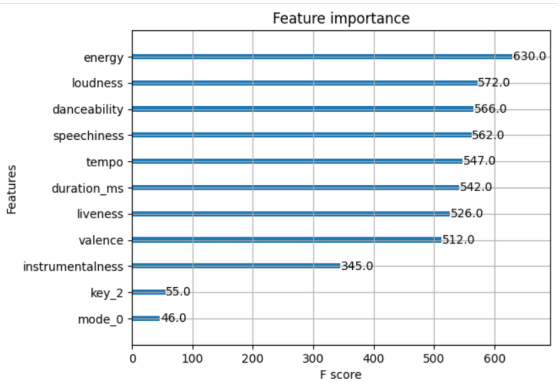
計算ROC曲線下面積，得AUC值為 0.881：

```
AUC = 0.8805883482607979
```

- XGBoost

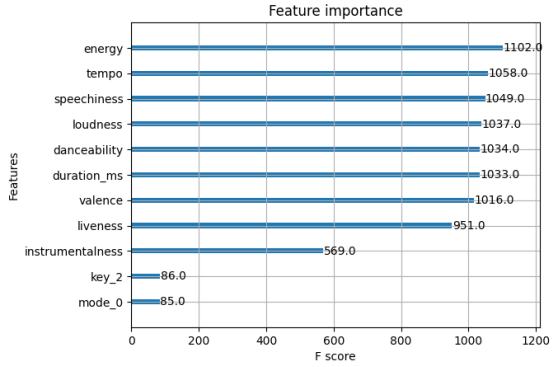
  Sklearn 介面分類器表現及其重要性特徵

```
Accuracy: 88.34 %
F1 score: 0.893
```



原生介面分類器表現及其重要性特徵

```
Accuracy: 88.96 %
F1 score: 0.899
```



由以上三組模型評估成績，觀察出**XGBoost**的原生介面表現最佳，因此選作為我們主要的機器學習模型使用。

捌、遇到的困難及解決方法

- Spotify API Client 設定與授權用戶

  解決方法：先查閱 Spotify Developer 手冊網站，再依指示進入 Spotify for Developers 的 Dashboard 建立App、取得 Client ID 及 Client Secret，接著進入 User Management 設定並管理使用者。

- 型別相容性問題

  解決方法：先用Pandas列出資料集中的各變項資料型態，比對API Documentation 以確認是否有可以直接捨棄的變項（如：track_id），接著檢查數值型態的變項中是否有類別變數，以及類別變數中是否為二元資料，查詢合適的型別轉換方法並實作，然後進行資料處理（取虛擬變數、正規化、填補空值、資料格式處理）。

- XGBoost Model 模型參數調整方式

  解決方法：原本嘗試進行 Grid 交叉驗證，但參數過多影響執行效率，因此也有嘗試隨機方式抽取，但效果不如手動調參的成效佳，故最後參數經由手動調參完成。

- 前後端整合設計

  解決方法：先以 tkinter 設計圖形使用者介面，再將我們原本的主程式改寫為兩個大函式（前者為依據使用者所輸入的歌手名稱蒐集歌曲；後者為使用模型建立推薦的歌單），但實作過程中，發現單機版的系統近用性不高且 tkinter 介面排版無法調整符合我們的審美觀，因此後來改以 Flask 設計使用者互動介面。

玖、心得感想

通過這個專案，我能實踐使用API進行數據存取和搜尋的過程，也能應用課堂所學的資料分析與視覺化技術、機器學習流程（訓練、預測、評估、調參）於解決生活上的問題，最後模型準確率達88%，實際使用系統後並分享給同學後，也都得到正面的回饋，令我加深對機器學習的興趣與熱忱。