音樂推薦系統

113213504 林千榆

目錄

你可以使用連結快速前往簡報中的不同頁面。

醒目提示文字，按一下工具列上的連結符號，然後選擇簡報中你要連結的頁面。

動機

資料處理

返回議程

title	artist	top genre	year	bpm	nrgy	dnce	dB
Hey, Soul Sister	Train	neo mellow	2010	97	89	67	
Love The Way You Lie	Eminem	detroit hip hop	2010	87	93	75	
TiK ToK	Kesha	dance pop	2010	120	84	76	
Bad Romance	Lady Gaga	dance pop	2010	119	92	70	
Just the Way You Are	Bruno Mars	pop	2010	109	84	64	
Baby	Justin Bieber	canadian pop	2010	65	86	73	
Dynamite	Taio Cruz	dance pop	2010	120	78	75	
Secrets	OneRepublic	dance pop	2010	148	76	52	

資料集

以spofity 的音樂為主 共10萬筆
80% 訓練集 20% 測試集

特徵

Nrgy	能量指數	Bpm	每分鐘節拍數
dnce	舞蹈性	Spch	語音成分
db	音量	pop	歌曲熱門程度
val	情緒正向值	dur	歌曲長度
acous	原音程度		

資料處理

匯入資料集

清理欄位集標題

選擇特徵

特徵標準化

利用pandas 匯入
編碼格式為：iso-8859-1
(因為資料含特殊字元）

返回議程

計算模型

```
kmeans = KMeans(n_clusters=10, random_state=
df['cluster'] = kmeans.fit_predict(X_scaled)

X_train, X_test, y_train, y_test = train_tes

tree_model = DecisionTreeClassifier(max_dept
tree_model.fit(X_train, y_train)

rf_model = RandomForestClassifier(n_estimato
rf_model.fit(X_train, y_train)
```

kmeans

將歌曲聚成10 群，無監督學習

決策樹

監督式學習，用來學習如何根據特徵預測歌曲類型（Cluster）。

隨機森林

提升預測穩定與準確度

最適群體處理
ELBOW METHOD

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} |p - mi|^2$$

用來選擇K means最佳群聚數
透過Within-cluster sum of squares計算每一群中的每一個資料點到群中心的距離， 來找出K 值

- SSE 代表集群的好壞，也就是所有資料的誤差
- 總共有 K 個群
- Ci 代表其中一群，也就是第i個群
- p 代表 Ci 中的資料點
- mi 代表該群心，也就是 Ci中所有資料的平均值

餘弦相似性
COSINE_SIMILARITY

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

通過測量兩個向量的夾角餘弦值來計算相似性

推薦函數

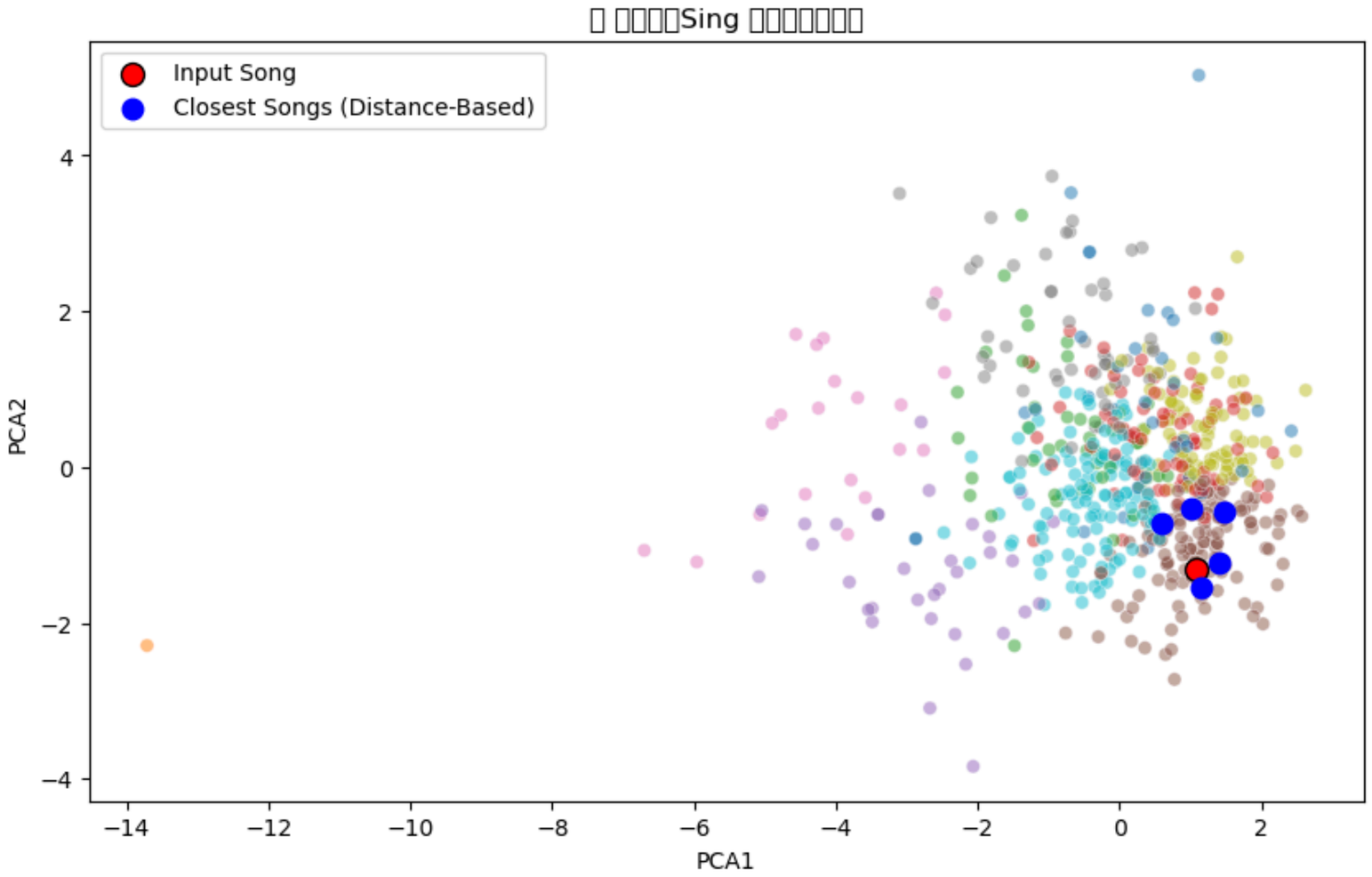
```python
def fusion_recommend(song_name, top_n=5):
    song_name = song_name.split(" (")[0]

    if song_name not in df['title'].values:
        artist_name = df[df['title'].str.contains(song_name, case=False, na=False)]['artist'].values
        if len(artist_name) > 0:
            artist_name = artist_name[0]
        else:
            artist_name = search_artist_online(song_name)  # 用 API 查找歌手名稱

        if artist_name:
            return df[df['artist'] == artist_name][['title', 'artist']].head(top_n)
        else:
            return None  # 如果找不到歌手，就回傳 None

    # 如果歌曲在資料庫中，則執行原本的推薦邏輯：
    idx = df[df['title'] == song_name].index[0]
    kmeans_cluster = df.loc[idx, 'cluster']
    tree_cluster = get_cluster(song_name, tree_model)
    rf_cluster = get_cluster(song_name, rf_model)

    recs = []
    recs += recommend_from_cluster(kmeans_cluster, song_name, 10)['title'].tolist()
    recs += recommend_from_cluster(tree_cluster, song_name, 10)['title'].tolist()
    recs += recommend_from_cluster(rf_cluster, song_name, 10)['title'].tolist()
    recs += get_similar_songs(song_name, 10)['title'].tolist()  # 加入相似度推薦

    counter = Counter(recs)
    common_titles = [title for title, _ in counter.most_common() if title != song_name]

    rec_df = df[df['title'].isin(common_titles)][['title', 'artist']].drop_duplicates()

    return rec_df.head(top_n)
```

融合推薦

- 透過三個候選集(每個十首歌曲) 並統計次數最多的歌曲
- 避免單一模型影響推薦效果

PCA

從九維降至兩維


Input Song
Closest Songs (Distance-Based)
PCA2
PCA1
4
2
0
−2
−4
−14
−12
−10
−8
−6
−4
−2
0
2

DEMO

音樂推薦系統 2009~2019

請輸入歌曲名稱：

Shape of You

推薦歌曲

- Hey, Soul Sister - Train
- TiK ToK - Kesha
- Baby - Justin Bieber
- Dynamite - Taio Cruz
- Only Girl - Rihanna

YouTube 播放