TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG.TPHCM
NHÓM NGÀNH TOÁN HỌC – TOÁN ỨNG DỤNG – TOÁN TIN

PYTHON CHO KHOA HỌC DỮ LIỆU

ĐỒ ÁN CUỐI KÌ

EDA IN SPOTIFY DATASET AND BUILD A MUSIC RECOMMENDATION SYSTEM

GIẢNG VIÊN PHỤ TRÁCH
Thầy Hà Văn Thảo

Thành phố Hồ Chí Minh, tháng 1, năm 2025

MỤC LỤC

NỘI DUNG

1.GIỚI THIỆU

1.1 Thông tin nhóm

- Tên nhóm: Nhóm 1 - Tuần 17 – C205 – Chiều
- Thành viên:
  - 22110092 – Nguyễn Thanh Kiên
  - 21110400 – Nguyễn Thế Phong
  - 21110277 – Nguyễn Thái Duy
  - 20110204 – Lâm Quang Khải

1.2 Nguyên tắc hoạt động nhóm

- Các thành viên trong nhóm đưa ra ý tưởng và đề tài, nhóm trưởng chọn lọc đề tài được đưa.
- Nhóm trưởng giao việc cho các thành viên, mọi người hoàn thành công việc được giao đúng hạn. Nếu không sẽ bị giảm điểm contributor.

1.3 Data

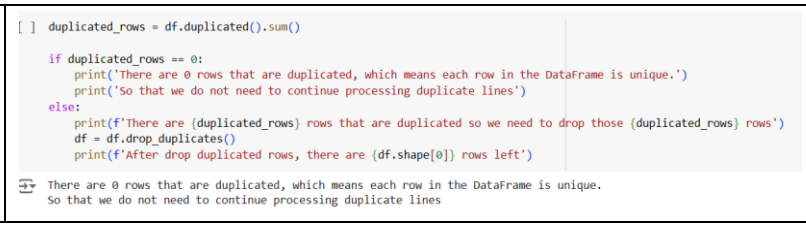
- Tập dữ liệu này được trích xuất từ nền tảng Spotify bằng thư viện Python "Spotipy", cho phép người dùng truy cập dữ liệu âm nhạc được cung cấp qua API. Tập dữ liệu được thu thập bao gồm khoảng 1 triệu bản nhạc với 19 đặc điểm từ năm 2000 đến năm 2023. Ngoài ra, dữ liệu còn có tổng cộng 61.445 nghệ sĩ độc đáo và 82 thể loại.
- Link: Spotify_1Million_Tracks

Audio Features	Description
Popularity	Track popularity (0 to 100)
Year	Year released (2000 to 2023)
Danceability	Track suitability for dancing (0.0 to 1.0)
Energy	The perceptual measure of intensity and activity (0.0 to 1.0)
Key	The key, the track is in (-1 to -11)
Loudness	Overall loudness of track in decibels (-60 to 0 dB)
Mode	Modality of the track (Major '1'/ Minor '0')
Speechiness	Presence of spoken words in the track
Acousticness	Confidence measure from 0 to 1 of whether the track is acoustic
Instrumentalness	Whether tracks contain vocals. (0.0 to 1.0)
Liveness	Presence of audience in the recording (0.0 – 1.0)
Valence	Musical positiveness (0.0 to 1.0)
Tempo	Tempo of the track in beats per minute (BPM)
Time_signature	Estimated time signature (3 to 7)
Duration_ms	Duration of track in milliseconds

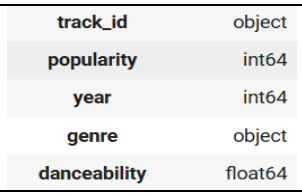
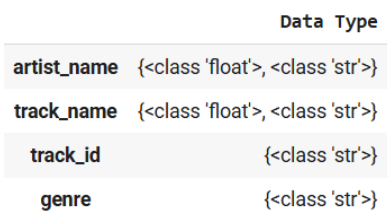
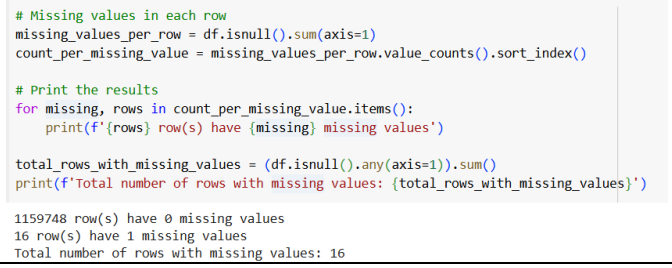
2. Khai phá Dữ liệu – EDA

2.1Một vài thông tin cơ bản

Số lượng bài hát	`ncols, nrows = df.shape`
	`print(f'Dataset has {ncols} rows and {nrows} columns')`
	
	`Dataset has 1159764 rows and 19 columns`
		Có hơn 1 triệu bài hát với 19 đặc điểm

Bài hát trùng lặp		Không có bài hát trùng lặp trong dữ liệu

2.2 Kiểu dữ liệu của mỗi cột

Kiểu dữ liệu		Các cột có rất nhiều kiểu dữ liệu khác nhau như obj, float, int,...
Kiểu con của Object		Có 2 kiểu dữ liệu không phù hợp, nó chứa cả kiểu "float" và "string". Do sự hiện diện của các giá trị NaN trong các cột này, vì NaN thuộc kiểu float
Sự đầy đủ của các giá trị trong mỗi hàng		Tất cả các hàng không có giá trị bị thiếu ngoại trừ 16 hàng có 1 giá trị bị thiếu. Ta sẽ xử lý những hàng cụ thể này trong phần tiếp theo.

2.3. Phân phối dữ liệu – Data Distribution

2.3.a Cột số

Đối với các cột chứa số, ta cần trích xuất và lưu chúng trong biến **numerical_cols. Sau đó, ta phải xử lý các** Missing Values Percentage bằng việc tính toán tỉ lệ phần trăm của nó.

	min	max	Missing Values	Missing Percentage
popularity	0.0	100.000	0	0.0
year	2000.0	2023.000	0	0.0
danceability	0.0	0.993	0	0.0
energy	0.0	1.000	0	0.0
key	0.0	11.000	0	0.0
loudness	-58.1	6.172	0	0.0
mode	0.0	1.000	0	0.0
speechiness	0.0	0.971	0	0.0
acousticness	0.0	0.996	0	0.0
instrumentalness	0.0	1.000	0	0.0
liveness	0.0	1.000	0	0.0
valence	0.0	1.000	0	0.0
tempo	0.0	249.993	0	0.0
duration_ms	2073.0	6000495.000	0	0.0
time_signature	0.0	5.000	0	0.0

```
dist_numerical_cols = numerical_cols.describe().T[['min', 'max']]
dist_numerical_cols['Missing Values'] = numerical_cols.isnull().sum()
dist_numerical_cols['Missing Percentage'] = (numerical_cols.isnull().mean() * 100).round(2)
# The number of -1 values in the 'key' column
dist_numerical_cols.loc['key', 'Missing Values'] = (df['key'] == -1).sum()
dist_numerical_cols
```

Nhưng ở cột key sẽ có giá trị -1 nếu không xác định được key của bài hát. Vì vậy, chúng ta sẽ tính số giá trị còn thiếu trong cột bằng cách đếm số giá trị -1 ở cột "key". Sau đó, chúng ta sẽ vẽ phân phối dữ liệu của Cột số và rút ra một số nhận xét.

Kết quả đầu ra cho thấy tất cả các cột số đều có tỷ lệ phần trăm giá trị bị thiếu là 0, cho biết rằng không có giá trị nào bị thiếu trong các cột này.

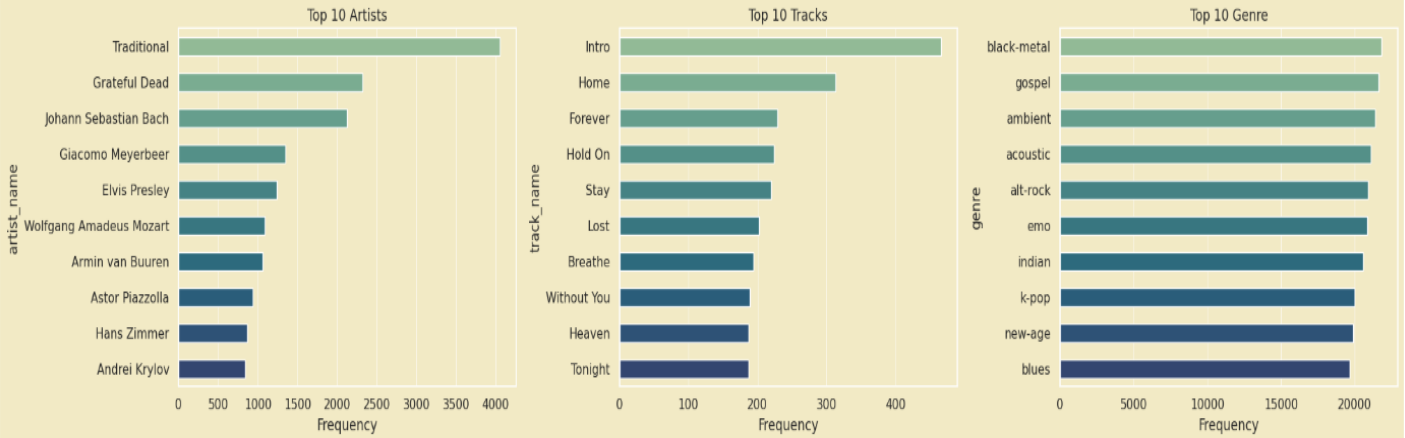
- Danceability, Tempo, Valence: Phân phối của danceability và valence gần với phân phối chuẩn.
- Loudness: Có phân phối lệch trái, chủ yếu ở mức từ -13 đến -8 dB.
- Speechiness, Acousticness, Instrumentalness, Liveness: Những đặc trưng này vẫn có phân phối lệch phải với giá trị thấp.
- Duration (duration_ms): Có phân phối lệch phải, với phần lớn bài hát dài trên 400.000ms (6 phút), một số dài đến 600.000ms (8 phút).
- Year: các bài hát phân phối đều từ năm 2000 đến năm 2023.

2.3.b Cột phân loại

Ta sẽ xử lý tương tự như cột số. Tạo biến Numerical_cols, kiểm tra Missing Values Percentage. Tuy vậy, ta phải xử lý bằng cách loại bỏ 16 giá trị NaN mà ta đã thấy ở phần trước. Ta sẽ vẽ ra top các nghệ sĩ và bài hát để đánh giá.

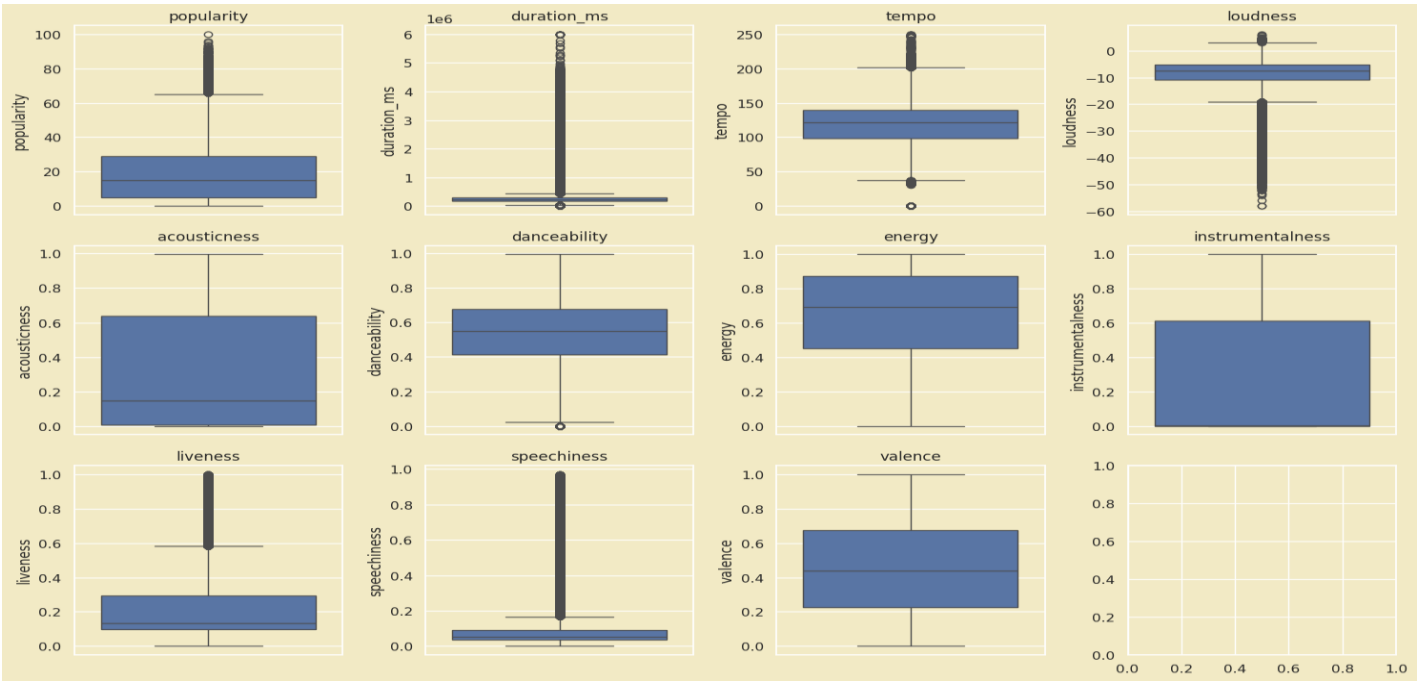
	artist_name	track_name	track_id	genre
256971	NaN	Cold	10eyeJsKDh26SKEBvSfuBG	black-metal
256972	NaN	Wither	6xz1O7tOw6Tdvzc3Ev1isc	black-metal
257024	NaN	Suffer	6hedSqpvIaW0GzrqJnq32D	black-metal
313028	NaN	A World, Dead and Gray	7shu4LrpMTsGwa8YotA1My	black-metal
313050	NaN	Hypoxic	3V8qpQnLVhu82ZvC4Ja8VG	black-metal
313063	NaN	Bed the Cold Earth	68w6nJMmA9Ui6zV7qXrwIt	black-metal
313117	NaN	Life Is Long Enough	1NJ8HTD5syI65EJmXUckxB	black-metal
313153	NaN	Corroded	4zcJ5p91PSG3vDsGJfENDd	black-metal
313174	NaN	Desiderate	6375ZIE3Pi8BuZOuy8f6g1	black-metal
313225	NaN	Where Life Should Be	0VCAJwhy4p3tVaGUSnaWGr	black-metal
369500	NaN	The Damp Chill of Life	1SanRTG8EZRb9U1V4NUSKp	black-metal
369521	NaN	It's Painless to Let Go	773j5fNWIjO0EWqAiX8Quo	black-metal
369578	NaN	You Did a Good Thing	7K0hyoMx2UZMf173FSUFnY	black-metal
369581	NaN	A Chance I'd Never Have	7ItLwpnDJNmnmjlx4hOC96	black-metal
369607	NaN	Cease	58NSB7DmCpMa7pAsfBCG5t	black-metal
879020	The Duskfall	NaN	2Q5cMgzptSupzvvWtZTyVg	swedish

16 cột có giá trị bị thiếu đều nằm ở 16 hàng cụ thể này. Ta sẽ tiến hành bỏ 16 hàng đó.



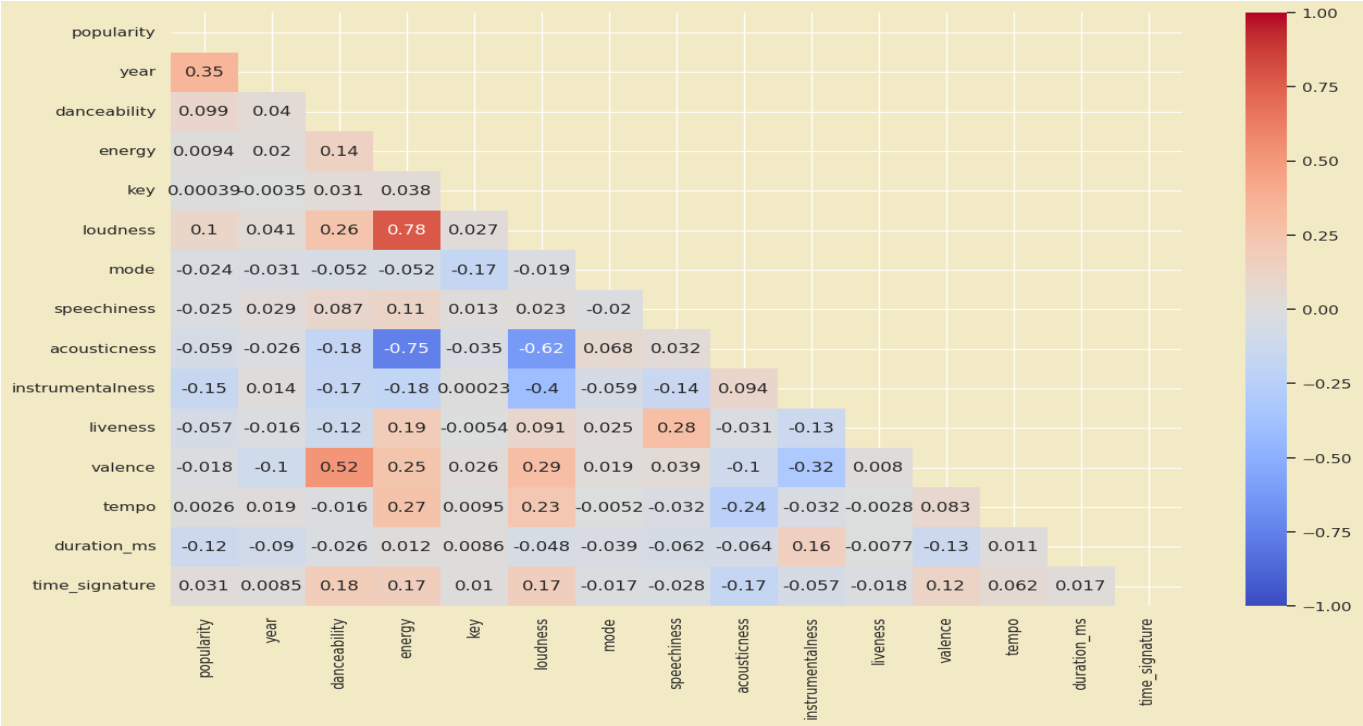
- Greateful Dead là Artists nổi tiếng nhất.
- Intro là bài hát được nghe nhiều nhất.
- 10 thể loại bài hát hay nhất đều là những thể loại phổ biến và chúng đều xuất hiện với tần suất như nhau (tần số = ~20.000).

Ta có thể quan sát thấy rằng sự phân bố cũng như phạm vi giá trị trong các cột số không có giá trị bất thường (abnormal values) nên chúng ta sẽ kiểm tra các giá trị ngoại lệ đối với một số cột. Ta sử dụng boxplot để trực quan hóa các ngoại lệ.



Nhận xét: Các thuộc tính **acousticness**, **danceability**, **energy**, **instrumentalness**, và **valence** không có giá trị ngoại lệ (outliers). Trong khi đó, các cột khác cho thấy có một lượng đáng kể các outliers, do chúng là các biến liên tục với phạm vi giá trị rộng, không tuân theo phân phối chuẩn và bị ảnh hưởng bởi nhiều yếu tố khác nhau của bài hát. Việc giữ lại các giá trị ngoại lệ này là hợp lý, vì chúng phản ánh sự đa dạng và phức tạp của tập dữ liệu, thể hiện tính chất đa dạng của các thuộc tính âm nhạc.

Ta tiếp tục vẽ Heatmap Correlation để tính toán mối tương quan giữa các cột số và trực quan hóa nó.



Nhận xét:

- **Tương quan cao giữa energy và loudness:** Hệ số tương quan là 0.78, cho thấy các bài hát có năng lượng cao thường đi kèm với âm lượng lớn.
- **Tương quan âm giữa acousticness và loudness (-0.62):** Cho thấy các bài hát có tính acoustic cao thường có âm lượng thấp, điều này dễ hiểu khi các bài hát mang phong cách acoustic thường yên tĩnh và nhẹ nhàng hơn.
- **Tương quan âm giữa acousticness và energy:** Hệ số -0.75 nhấn mạnh rằng các bài hát có tính acoustic cao thường có năng lượng thấp.
- **Tương quan giữa danceability và valence (0.52):** Hệ số 0.52 cho thấy các bài hát có cảm xúc tích cực thường dễ nhảy hơn.
- **Tương quan âm giữa instrumentalness và loudness (-0.4):** Cho thấy rằng các bài hát không lời thường có âm lượng thấp hơn.
- **Tương quan giữa tempo và energy**: Hệ số 0.27 cho thấy các bài hát có nhịp độ cao hơn có xu hướng có năng lượng cao hơn, nhưng mức độ tương quan không mạnh.
- **Popularity:** Có mối tương quan thấp với tất cả các biến số khác, do đây là thước đo chủ quan bị ảnh hưởng bởi nhiều yếu tố như độ nổi tiếng của nghệ sĩ, ngày phát hành, và thể loại.

3 Tiến sâu hơn vào bài toán

Để thiết kế một hệ thống gợi ý nhạc hiệu quả, cần hiểu rõ mối quan hệ giữa "thể loại nhạc" (genre), "mức độ phổ biến" (popularity), và các yếu tố như "nghệ sĩ" hay "tâm trạng người nghe". Điều này giúp tối ưu hóa trải nghiệm người dùng và nâng cao hiệu suất hệ thống. Hai câu hỏi chính cần xem xét:

a.Thể loại nhạc ảnh hưởng đến mức độ phổ biến thế nào?

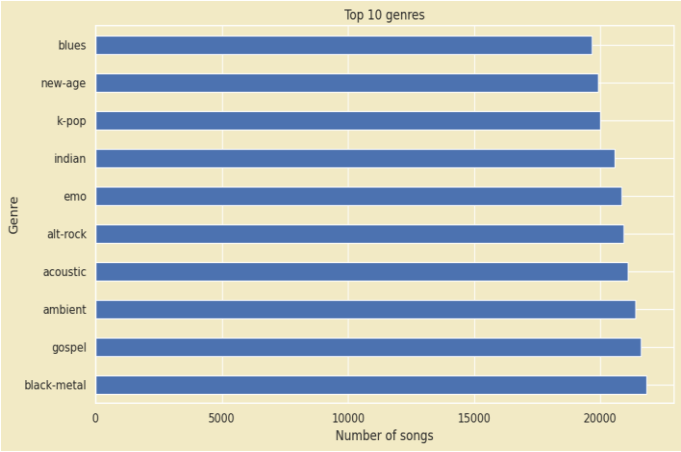
Mỗi thể loại nhạc thu hút một nhóm khán giả với sở thích riêng. Ví dụ, nhạc pop dễ phổ biến do giai điệu dễ nghe, trong khi nhạc cổ điển có lượng người nghe trung thành nhưng quy mô nhỏ hơn. Phân tích dữ liệu từ nền tảng như Spotify giúp làm rõ mối quan hệ giữa thể loại và mức độ phổ biến dựa trên tần suất nghe, lượng người hâm mộ trung thành, và mức độ lan tỏa.

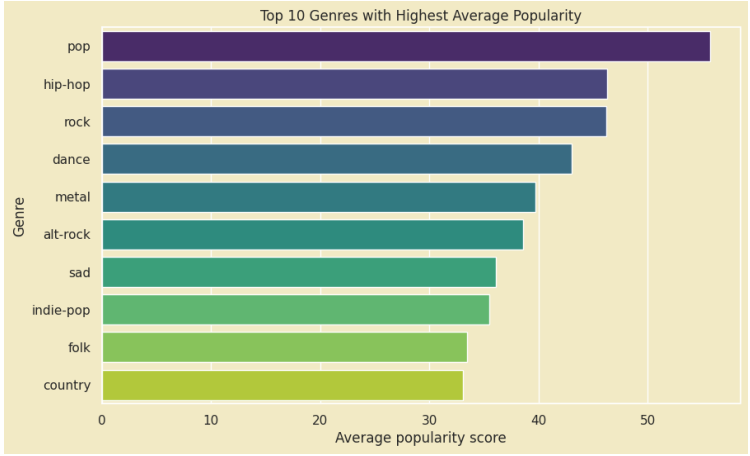
b.Nghệ sĩ nên chọn thể loại nhạc ra sao?

Nghệ sĩ cần cân nhắc phong cách cá nhân, đối tượng mục tiêu, và xu hướng thị trường để chọn thể loại phù hợp. Hệ thống gợi ý có thể hỗ trợ họ nắm bắt xu hướng, tạo sản phẩm dễ tiếp cận và định hình phong cách lâu dài, đồng thời tăng kết nối với người nghe.

3.1 Thể loại nhạc ảnh hưởng đến mức độ phổ biến thế nào?

Để xử lý câu hỏi này, ta phân tích đặc điểm liên quan đến thể loại nhạc: Số lượng bài hát cho mỗi thể loại. Điểm phổ biến trung bình (average popularity) cho từng thể loại. Thể loại nằm trong top ~110 bài hát phổ biến nhất. Tỷ lệ các bài hát trực tiếp cho từng thể loại. So sánh mối tương quan giữa loundness và energy ở top 3 thể loại nhạc được yêu thích nhất.

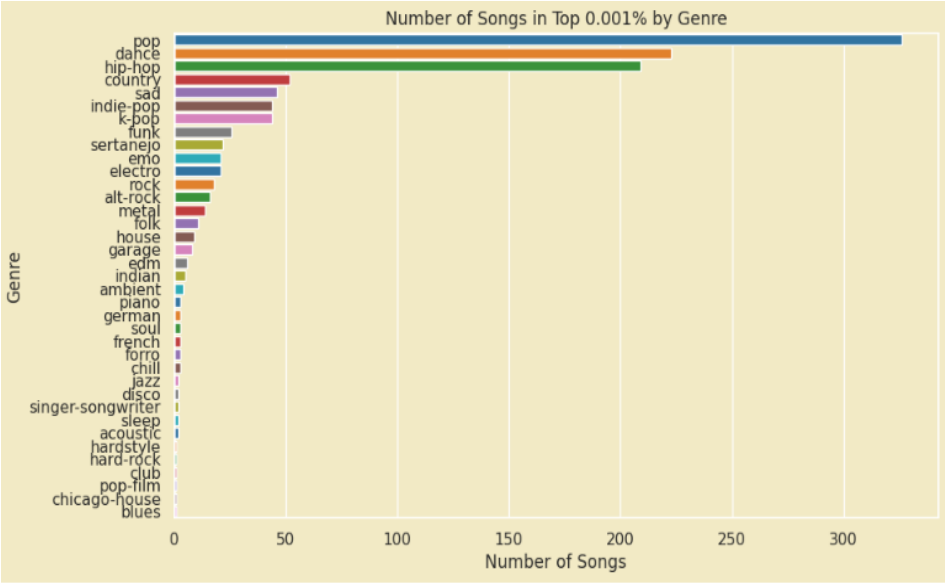




Dựa vào biểu đồ 1 có thể thấy top 10 thể loại có số lượng bài hát nhiều nhất đều trên 15.000 bài. Ở biểu đồ 2, ta sẽ xem thể loại nào có average popularity cao nhất. Dựa trên biểu đồ, ta nhận thấy điểm phổ biến trung bình cao nhất cho một thể loại là khoảng gần 57, cao hơn đáng kể so với các thể loại còn lại trong top 10.

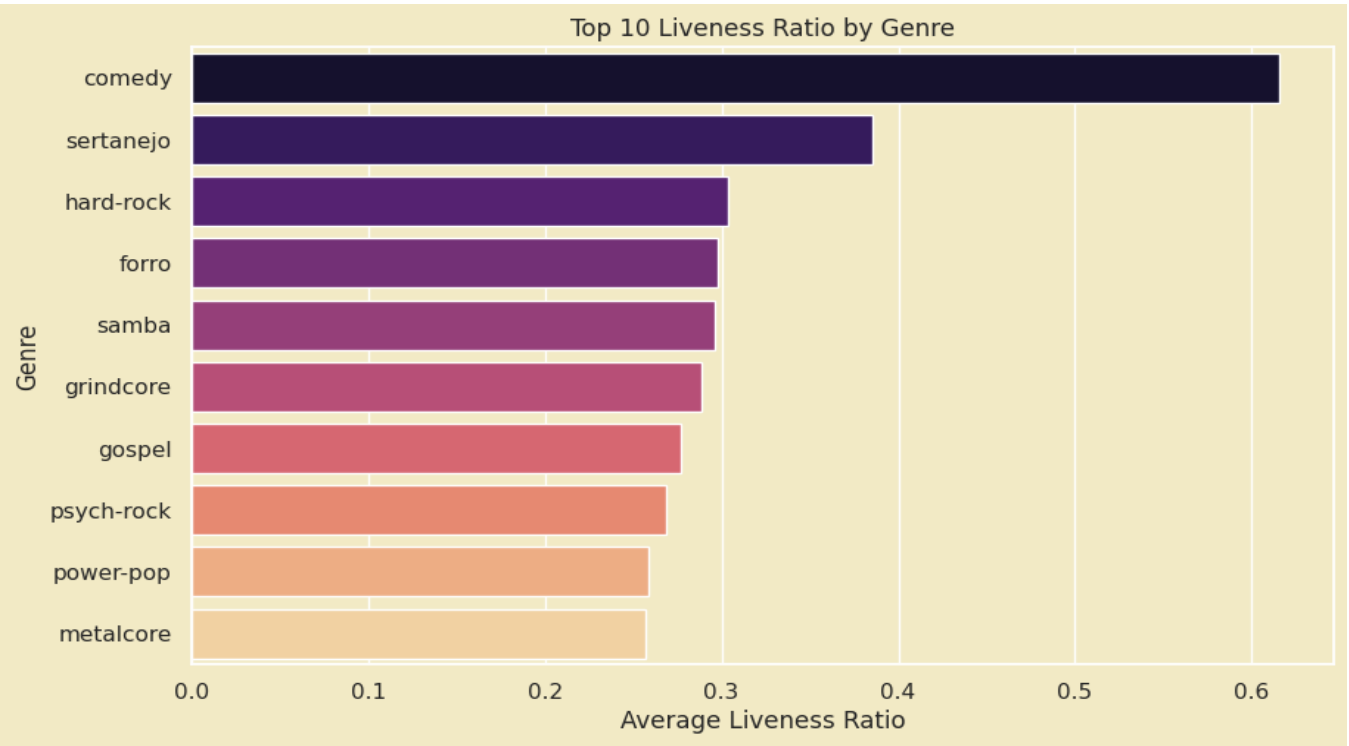
```
df_sorted = df.sort_values(by='popularity', ascending=False)

top_10_percent = df_sorted.head(int(0.001 * len(df_sorted)))
colors = sns.color_palette("tab10", len(top_10_percent))

top_genres_count = top_10_percent.groupby('genre').size().reset_index(name='count')
top_genres_count = top_genres_count.sort_values(by='count', ascending=False)

most_common_genre = top_genres_count.loc[top_genres_count['count'].idxmax()]

plt.figure(figsize=(10, 6))
sns.barplot(x='count', y='genre', data=top_genres_count, palette = colors)
plt.title('Number of Songs in Top 0.001% by Genre')
plt.xlabel('Number of Songs')
plt.ylabel('Genre')
plt.show()

print(f"The genre with the most songs in the top 0.001% is {most_common_genre['genre']} with {most_common_genre['count']} songs.")
```

- df.sort_values: Sắp xếp DataFrame df theo cột popularity theo thứ tự giảm dần.
- head(int(0.001 * len(df_sorted))): Lấy 0.001% (tức là 1 phần nghìn) bài hát từ DataFrame đã sắp xếp.
- groupby('genre').size(): Nhóm các bài hát theo thể loại và đếm số lượng bài hát trong mỗi thể loại.
- reset_index(name='count'): Đặt lại chỉ số và đổi tên cột đếm thành count.
- Sắp xếp DataFrame top_genres_count theo cột count theo thứ tự giảm dần.
- idxmax(): Tìm chỉ số của thể loại có số lượng bài hát nhiều nhất.
- loc[...]: Lấy thông tin của thể loại đó từ DataFrame.



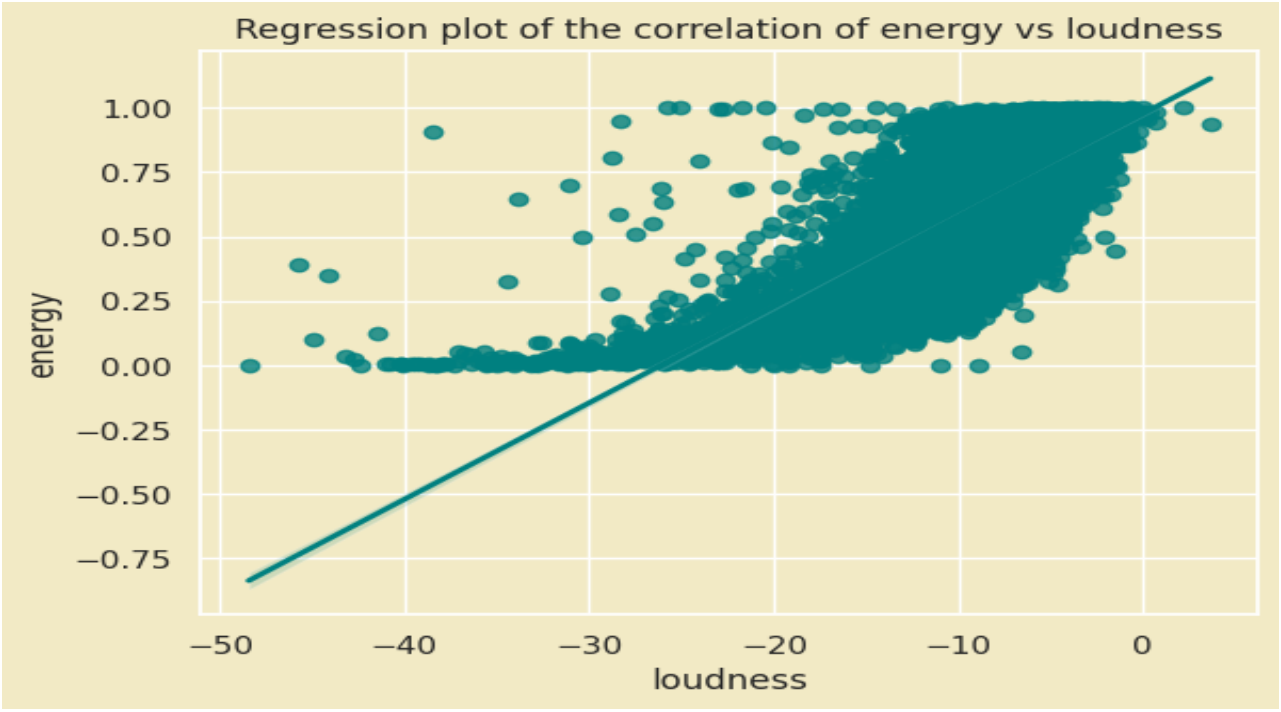
Tiếp tục, ta tìm xem thể loại nào có số lượng bài hát cao nhất trong top 0,001% bài hát phổ biến nhất? Mục tiêu của việc phân tích là xác định thể loại chiếm ưu thế trong số các bài hát nổi tiếng được xếp hạng hàng đầu. Sắp xếp DataFrame theo cột 'popularity' theo thứ tự giảm dần. Chọn số hàng tương ứng với 0,001% tổng số hàng trong DataFrame đã sắp xếp.

Tương tự đối với tỷ lệ bài hát trực tiếp (live) cao nhất.

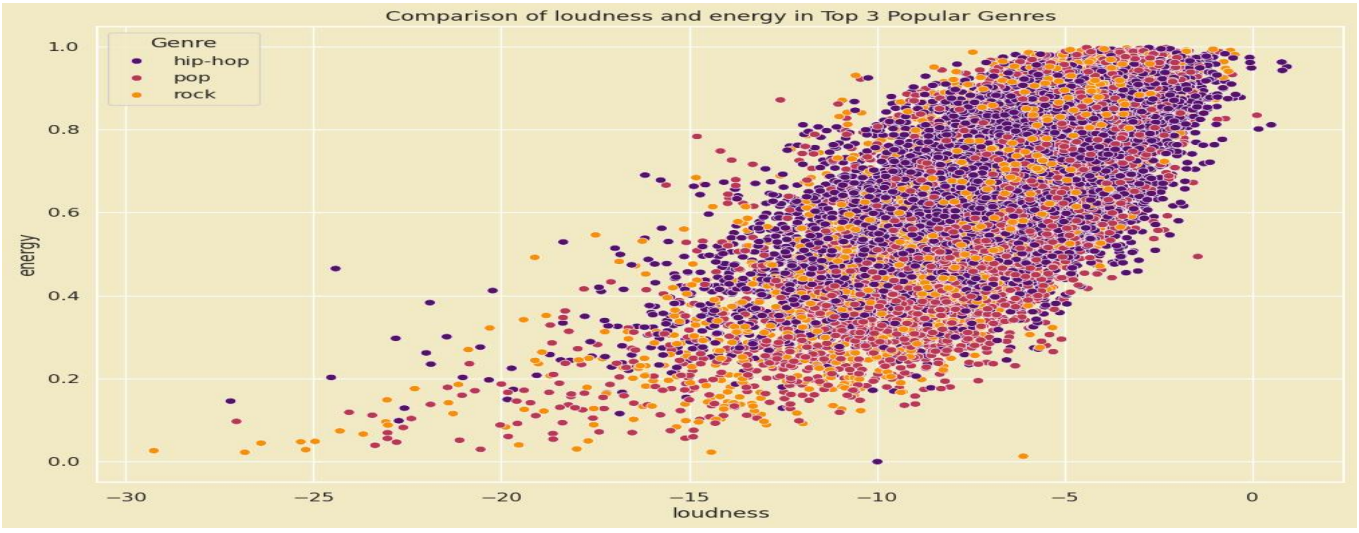


Thể loại comedy có tỷ lệ live cao nhất do sự tương tác trực tiếp và phản ứng của khán giả là yếu tố quan trọng. Các buổi biểu diễn hài mang đến cơ hội tương tác, ngẫu hứng và tạo không khí sôi động. Tỷ lệ bài hát live cao phản ánh sở thích khán giả với những trải nghiệm gần gũi, độc đáo, nơi nghệ sĩ có tự do sáng tạo và kết nối sâu sắc hơn với khán giả.

Dựa vào phần tương quan, ta nhận thấy mối tương quan cao nhất là giữa độ ồn và năng lượng



Ta tiếp tục tìm xem tương quan giữa loudness và energy ở top 3 thể loại có average popularity cao nhất ?



Biểu đồ cho thấy mối tương quan đáng kể giữa độ ồn và năng lượng ở 3 thể loại âm nhạc phổ biến nhất. Bài hát có âm lượng lớn thường mang lại cảm giác mạnh mẽ, năng động, và giàu năng lượng do mức âm thanh và tính năng động cao. Những bản nhạc sôi động, cảm xúc mạnh thường thu hút khán giả, giúp chúng trở nên phổ biến hơn.

Sau khi phân tích và hình dung kết quả, có thể rút ra một số kết luận như sau:
- **4 thể loại có AVG Popularity cao nhất**: Pop, Hiphop, Rock và Dance.
- **Thể loại nhạc pop**: Nổi bật trong top 0.001% ca khúc được yêu thích nhất, chiếm khoảng 1/5 số bài hát.
- **Các thể loại biểu diễn trực tiếp hàng đầu**: Comedy, Sertanejo, Hard-rock và Forno.
- **Mối tương quan giữa loudness và energy**: Rất cao, cho thấy sự kết nối giữa âm lượng và năng lượng của bài hát.

Tiếp đến ta phân tích mức độ phổ biến:
- **Tác động của thể loại**: Thể loại nhạc pop vượt trội hơn các thể loại khác, đặc biệt ở các ca khúc đứng đầu.
- **Sở thích của khán giả**: Điểm popularity score phản ánh sở thích, cho thấy khán giả thích các buổi live tràn đầy năng lượng, như ở thể loại nhạc pop.
- **Quan điểm nghệ sĩ**: Một số nghệ sĩ có thể mong muốn nổi tiếng hơn, trong khi những người khác chọn theo sở thích cá nhân hơn xu hướng.

3.2 Nghệ sĩ nên lựa chọn thể loại âm nhạc như thế nào?

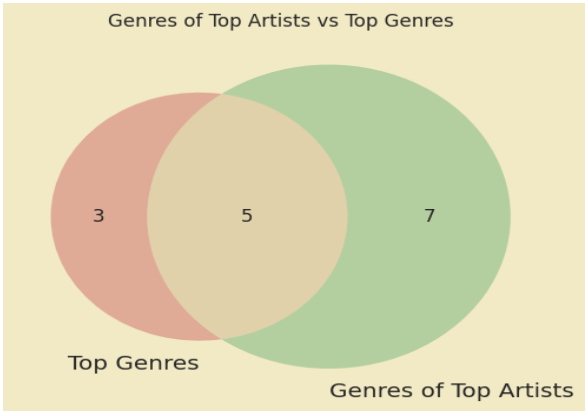
Chúng ta sẽ chia câu hỏi thành 2 câu hỏi nhỏ: Lựa chọn giữa bản sắc cá nhân hay xu hướng, nghệ sĩ nên chọn thể loại họ thích hay chọn những gì phổ biến? Tập trung một loại hay đa dạng hóa, liệu có tốt hơn không nếu nghệ sĩ chỉ bám vào một vài thể loại hoặc mở rộng phạm vi thể loại của họ?

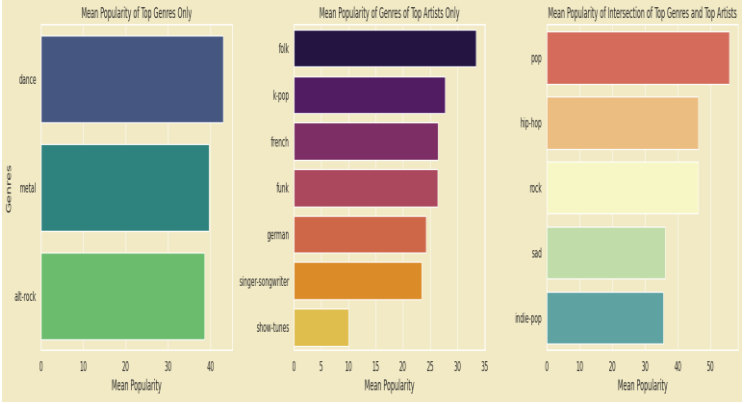
Bắt đầu bằng cách thực hiện một số bước tiền xử lý để chuẩn bị dữ liệu cho phân tích. Ta trích xuất:

Top Genres	`top_genres = df.groupby('genre')['popularity'].mean()`
	`top_genres = top_genres[top_genres >= 35].sort_values(ascending=False)`
	`top_genres[:10]`
		Các thể loại phổ biến nhất trong tập dữ liệu tính theo mức độ phổ biến trung bình của các bài hát trong đó.

Top Artists:	`top_artists = copy_df.groupby('artist_name')['popularity'].mean()`
	`top_artists = top_artists[top_artists >= 70].sort_values(ascending=False)`
	`top_artists[:10]`
		Các nghệ sĩ nổi tiếng nhất trong tập dữ liệu tính theo mức độ phổ biến trung bình của các bài hát của họ.

Genres of Top Artists	`top_artists = top_artists.index.tolist()`
	`genres_artists = df[df['artist_name'].isin(top_artists)]['genre'].unique().tolist()`
	`genres_artists[:10]`
		Các thể loại được thể hiện trong bài hát của nghệ sĩ hàng đầu





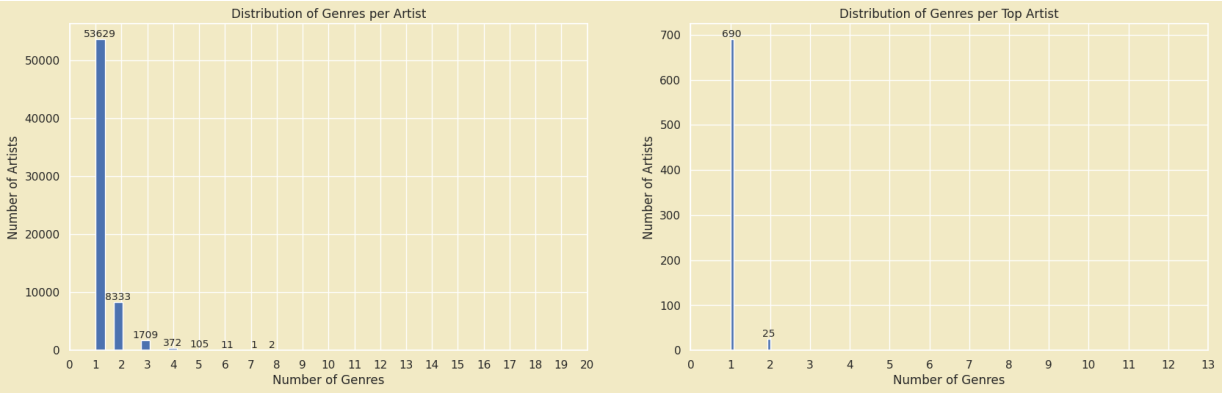
Biểu đồ 1 cho thấy sự trùng lặp giữa các thể loại âm nhạc phổ biến và lựa chọn của nghệ sĩ nổi tiếng. Trong số 8 thể loại, 5 thể loại được cả khán giả yêu thích và nghệ sĩ đam mê. Họ cũng khám phá 7 thể loại khác, thể hiện sự sáng tạo cá nhân.

Biểu đồ 2 chỉ ra rằng 3 thể loại âm nhạc phổ biến nhất là dance, metal và alt-rock, với mức độ phổ biến trung bình trên 38, cho thấy sự yêu thích mạnh mẽ. Mặc dù các thể loại gắn liền với sở thích của nghệ sĩ nổi tiếng được yêu thích, nhưng lại ít phổ biến hơn do không được biết đến rộng rãi. Các thể loại khác như pop, hip-hop, rock, sad và indie-pop có mức độ phổ biến trung bình trên 35 nhờ sức hấp dẫn và sự lựa chọn của nghệ sĩ nổi tiếng.

Phân tích Mức độ Phổ biến của Nghệ sĩ

Câu hỏi nhỏ thứ 2 sẽ phân tích mức độ phổ biến của các nghệ sĩ dựa trên phạm vi thể loại. So sánh giữa các nghệ sĩ có phạm vi thể loại hẹp và rộng nhằm đưa ra những hiểu biết và kết luận:

- **Phạm vi hẹp**: Nghệ sĩ có ít thể loại hơn so với các nghệ sĩ hàng đầu.
- **Phạm vi rộng**: Nghệ sĩ có phạm vi thể loại lớn hơn hoặc bằng với các nghệ sĩ hàng đầu.

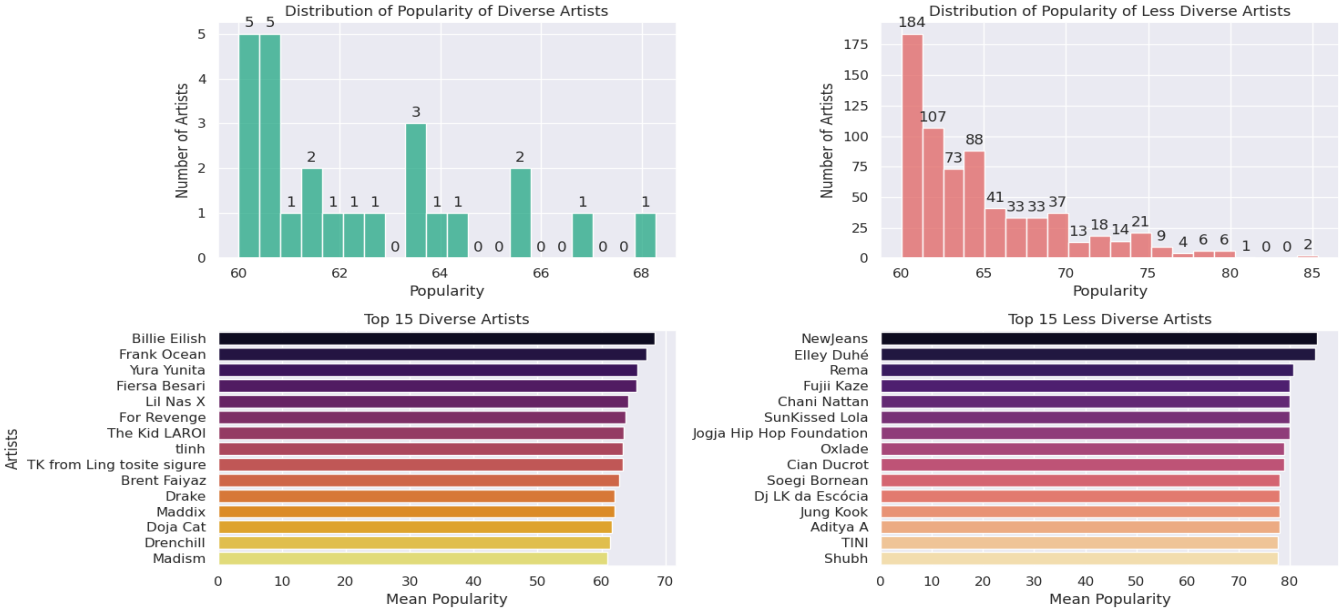


Trong toàn bộ tập dữ liệu: Phần lớn các nghệ sĩ cover 1-3 thể loại âm nhạc. Một số nghệ sĩ thể hiện sự đa dạng hơn, từ 4 đến 8 thể loại âm nhạc. Không có nghệ sĩ đi sâu vào 10 thể loại trở lên. Đối với top Artist: Các nghệ sĩ cũng bao gồm 1 hoặc 2 thể loại âm nhạc, phù hợp chặt chẽ với tập dữ liệu tổng thể. Vì vậy, có thể coi những nghệ sĩ có 1 thể loại trở lên là những nghệ sĩ có phạm vi đa dạng.

Tạo 4 biểu đồ để hình dung mức độ phổ biến của các nghệ sĩ có phạm vi thể loại hẹp và những nghệ sĩ có nhiều thể loại.

- 2 biểu đồ Histograms để trực quan hóa sự phân bố điểm phổ biến của các nghệ sĩ thuộc thể loại hẹp và những nghệ sĩ có nhiều thể loại rộng.
- 2 biểu đồ Bar charts để trực quan hóa điểm phổ biến trung bình của các nghệ sĩ hàng đầu với phạm vi thể loại hẹp và những nghệ sĩ có phạm vi thể loại rộng.

```
fig, axes = plt.subplots(2, 2, figsize=(15, 8))
sns.set(rc={"axes.facecolor":"#F2EAC5","figure.facecolor":"#F2EAC5"})
color_palettes = ['viridis', 'plasma', 'inferno', 'magma']

sns.histplot(diverse_artists, bins=20, ax=axes[0, 0], color=sns.color_palette(color_palettes[0], as_cmap=True)(0.6))
axes[0, 0].set_title('Distribution of Popularity of Diverse Artists')
axes[0, 0].set_xlabel('Popularity')
axes[0, 0].set_ylabel('Number of Artists')
axes[0, 0].bar_label(axes[0, 0].containers[0], padding=3,)

sns.histplot(less_diverse_artists, bins=20, ax=axes[0, 1], color=sns.color_palette(color_palettes[1], as_cmap=True)(0.6))
axes[0, 1].set_title('Distribution of Popularity of Less Diverse Artists')
axes[0, 1].set_xlabel('Popularity')
axes[0, 1].set_ylabel('Number of Artists')
axes[0, 1].bar_label(axes[0, 1].containers[0], padding=3)

sns.barplot(x=diverse_artists.sort_values(ascending=False).head(15).values, y=diverse_artists.sort_values(ascending=False)
.head(15).index, ax=axes[1, 0], palette=sns.color_palette(color_palettes[2], 15) )
axes[1, 0].set_title('Top 15 Diverse Artists')
axes[1, 0].set_xlabel('Mean Popularity')
axes[1, 0].set_ylabel('Artists')

sns.barplot(x=less_diverse_artists.sort_values(ascending=False).head(15).values, y=less_diverse_artists.sort_values(ascending=False).
            head(15).index, ax=axes[1, 1],  palette=sns.color_palette(color_palettes[3], 15))
axes[1, 1].set_title('Top 15 Less Diverse Artists')
axes[1, 1].set_xlabel('Mean Popularity')
axes[1, 1].set_ylabel('')

plt.tight_layout()
plt.show()
```



Mức độ phổ biến giữa các nghệ sĩ có sự khác biệt rõ rệt, cho thấy việc khám phá nhiều thể loại âm nhạc ảnh hưởng lớn đến sự nổi tiếng. Các nghệ sĩ như Billie Eilish, Drake, và Lil Nas X thuộc nhóm đa dạng và nổi tiếng. Mối tương quan cho thấy rằng theo đuổi nhiều thể loại có thể giúp tăng mức độ nổi tiếng nhờ sự đa dạng và cởi mở của khán giả. Taylor Swift là ví dụ điển hình với sự đa dạng trong thể loại âm nhạc.

Tuy nhiên, nghệ sĩ theo đuổi một thể loại cụ thể vẫn có thể thành công nếu họ nổi bật trong thể loại đó. Lựa chọn thể loại âm nhạc là một quyết định quan trọng, ảnh hưởng đến sự nghiệp và phong cách cá nhân. Nghệ sĩ nên cân nhắc sở thích cá nhân, xu hướng thị trường, và đánh giá kỹ năng để lựa chọn thể loại phù hợp.

Mặc dù việc khám phá nhiều thể loại không đảm bảo thành công lớn, nghệ sĩ cần tránh thay đổi thể loại quá thường xuyên để giữ vững sự ủng hộ từ người hâm mộ. Lựa chọn thể loại là hành trình cá nhân, kết hợp giữa đam mê và chiến lược phát triển sự nghiệp.

4 Xây dựng Music Recommendation System

Trong phần quan trọng này, nhóm chúng em sẽ làm hệ thống theo 2 cách gợi ý: Music Recommendation System After Listening To A Song và Music Recommendation System Using Listening History.

4.1 Music Recommendation System After Listening To A Song

Làm cách nào chúng ta có thể đề xuất bài hát dựa trên sở thích nghe hiện tại của người dùng, hay nói đơn giản hơn là, sau khi nghe 1 bài hát, làm sao để nghe được 1 bài hát tiếp theo có nét tương đồng với bài hát đã nghe trước đó ? Nhóm em sử dụng khoảng cách Euclide làm nền tảng:

- Sử dụng khoảng cách Euclide để đo độ tương tự giữa các bài hát. Khoảng cách Euclide càng nhỏ thì hai bài hát càng giống nhau xét từ góc độ các đặc điểm được chọn.

- Chọn K bài hát có khoảng cách nhỏ nhất. Đây là những bài hát được hệ thống đánh giá cao về độ tương đồng với bài hát người dùng yêu thích.

- Trích xuất thông tin như tên bài hát và nghệ sĩ từ các bài hát đã chọn và trả lại thông tin này dưới dạng đề xuất cho người dùng.

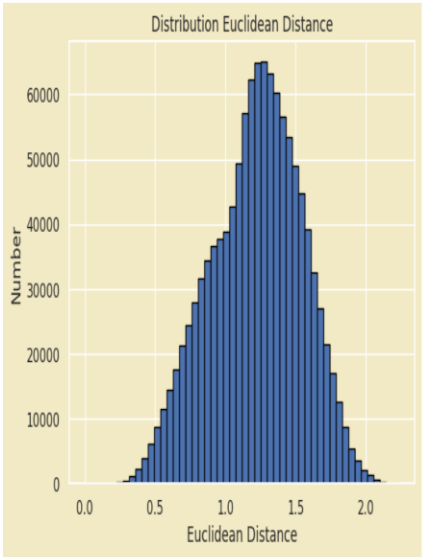
Đầu tiên, tụi em xóa cột 'time_signature' và 'key' là do tính chất phân loại của chúng. Sau đó, sử dụng MinMaxScaler để chuẩn hóa các tính năng số trong DataFrame. Tiếp tục, tạo DataFrame mới, data_norm, từ dữ liệu đã chuẩn hóa và đặt cột track_id index.

```
from sklearn import preprocessing
scaler = preprocessing.MinMaxScaler()

numerical_cols = df.select_dtypes(include=np.number).columns
data_norm = pd.DataFrame(scaler.fit_transform(df[numerical_cols]), columns=numerical_cols, index=df['track_id'])
```

Tụi em xác định bản nhạc mà người dùng đã nghe gần đây bằng việc truy xuất track_id tương ứng với bài hát được đề cập, giả sử người dùng gần đây đã nghe bài hát "nếu lúc đó".

```
trackNameListened = "nếu lúc đó"
track_id = df[(df['track_name'] == trackNameListened)][['track_id']]
track_id = track_id.values[0][0]

target_track = list(data_norm.loc[track_id])
```

Ở phần Khoảng cách Euclide là nền tảng cho bài toán, Sử dụng vòng lặp for để lặp qua tất cả các hàng trong data_norm. Ở mỗi lần lặp, tính khoảng cách Euclide giữa vectơ đặc trưng của obj rãnh được xem xét và vectơ đặc trưng của target_track. Các kết quả được lưu trữ vào cột Euclide của DataFrame .




```
data_result = pd.DataFrame()
data_result['euclidean'] = [distance.euclidean(obj, target_track) for index, obj in data_norm.iterrows()]
data_result['track_id'] = data_norm.index
```

Các cụm (cluster) trong biểu đồ có thể chỉ ra các nhóm bài hát giống nhau hơn.
Cuối cùng, tụi em trích xuất thông tin các bản nhạc được đề xuất. Lặp lại qua các bản nhạc được đề xuất, trích xuất thông tin về tên bài hát và nghệ sĩ từ DataFrame gốc và lưu nó vào DataFrame track_list.

Chạy hệ thống

Đề xuất 5 bài nhạc hàng đầu cho người dùng theo thứ tự độ giống nhau giảm dần, dựa trên khoảng cách Euclide được tính toán.

```
data_init = df.set_index(df.loc[:, 'track_id'])
track_list = pd.DataFrame()
for i in list(data_rec.loc[:, 'track_id']):
    if i in list(df.loc[:, 'track_id']):
        track_info = data_init.loc[[i], ['track_name', 'artist_name']]
        track_list = pd.concat([track_list, track_info], ignore_index=True)
```

```
recomended = track_list.values.tolist()
print(f"""You've just listened:  \n \t - {recomended[0][0]} - {recomended[0][1]}
Now you may listen :
\n \t - '{recomended[1][0]} - {recomended[1][1]}'
Or any of:
\n \t - '{recomended[2][0]} - {recomended[2][1]}'
\n \t - '{recomended[3][0]} - {recomended[3][1]}'
\n \t - '{recomended[4][0]} - {recomended[4][1]}'
\n \t - '{recomended[5][0]} - {recomended[5][1]}'  """)
```

```
You've just listened:
         - nếu lúc đó - tlinh
Now you may listen :

         - 'Howl - Elderbrook'
Or any of:

         - 'Lost Without You (with Dean Lewis) - Kygo'

         - 'Blindsided - Kelsea Ballerini'

         - 'Living In A Haze - Milky Chance'

         - 'Money Isn't Real - Jordan Davis'
```

Nhược điểm của cách làm này là chúng ta không thể nào có được những bài hát gợi ý tiếp theo mà ta muốn nghe theo sở thích của ta, mà chỉ có những bài hát được nghe giống với bài hát trước mà ta được nghe.

4.2 Music Recommendation System Using Listening History

Hệ thống đề xuất nhạc bằng cách sử dụng tập dữ liệu Spotify từ Kaggle. Hệ thống được xây dựng bằng các kỹ thuật Học máy để đề xuất bài hát cho người dùng dựa trên lịch sử nghe và sở thích của họ, nhằm mục đích dự đoán khả năng người dùng sẽ thích một bài hát. Bằng cách phân tích lịch sử bài hát trước đây của người dùng và các thuộc tính của bản nhạc, hệ thống sẽ tạo danh sách các bản nhạc được đề xuất.

Đầu tiên, ta tạo trải nghiệm được cá nhân hóa cho người dùng bằng cách đề xuất các bản nhạc dựa trên sở thích và thói quen nghe nhạc của từng cá nhân.

- **Pipeline**: Là một công cụ trong scikit-learn cho phép kết hợp nhiều bước xử lý dữ liệu và mô hình hóa trong một quy trình duy nhất.
- **StandardScaler()**: Bước đầu tiên trong pipeline là chuẩn hóa dữ liệu. StandardScaler sẽ biến đổi các đặc trưng để có trung bình bằng 0 và độ lệch chuẩn bằng 1. Điều này giúp cải thiện hiệu suất của thuật toán phân cụm.

```
song_cluster_pipeline = Pipeline([('scaler', StandardScaler()),
                                  ('kmeans', KMeans(n_clusters=30,
                                   verbose=False))
                                 ], verbose=False)

X = df.select_dtypes(np.number)
song_cluster_pipeline = song_cluster_pipeline.fit(X)
```

- **KMeans(n_clusters=30, verbose=False)**: Bước thứ hai là áp dụng thuật toán KMeans để phân cụm dữ liệu thành 30 nhóm (n_clusters=30). Tham số verbose=False chỉ định rằng không cần in ra thông tin chi tiết trong quá trình chạy.

- **df.select_dtypes(np.number)**: Dòng này chọn các cột trong DataFrame df mà có kiểu dữ liệu số. Dữ liệu số này sẽ được sử dụng để phân cụm.
- **fit(X)**: Phương thức fit sẽ huấn luyện pipeline với dữ liệu đã chọn (X). Quy trình chuẩn hóa dữ liệu và sau đó áp dụng KMeans để tìm các nhóm trong dữ liệu sẽ được thực hiện ở đây.

```
def get_song_data(song, spotify_data):
    try:
        song_data = spotify_data[(spotify_data['track_name'] == song['name']) &
                                 (spotify_data['year'] == song['year'])].iloc[0]
        return song_data
    except IndexError:
        return None
```

- Mục đích: Hàm này tìm kiếm thông tin về một bài hát cụ thể trong spotify_data.
- Cách hoạt động:
- Sử dụng điều kiện để lọc spotify_data theo tên bài hát và năm.
- Nếu tìm thấy, hàm trả về thông tin của bài hát đầu tiên (iloc[0]).
- Nếu không tìm thấy hàm trả về None.

- **Mục đích**: Tính toán vector trung bình cho một danh sách bài hát.
- **Cách hoạt động**:
  - Tạo danh sách *song_vectors* để lưu trữ vector của các bài hát.
  - Duyệt qua từng bài hát trong *song_list:*
  - Sử dụng *get_song_data* để lấy dữ liệu bài hát.

```
def get_mean_vector(song_list, spotify_data):
    song_vectors = []
    for song in song_list:
        song_data = get_song_data(song, spotify_data)
        if song_data is None:
            print('Warning: {} does not exist in Spotify or in database'.format(song['name']))
            continue
        song_vector = song_data[numerical_cols].values
        song_vectors.append(song_vector)

    song_matrix = np.array(list(song_vectors))
    return np.mean(song_matrix, axis=0)
```

- Nếu không tìm thấy bài hát, in ra cảnh báo và tiếp tục với bài hát tiếp theo.
- Lấy vector từ các cột số (được xác định bởi numerical_cols) và thêm vào danh sách song_vectors.
- Chuyển đổi danh sách thành mảng NumPy và tính trung bình theo chiều 0 (theo hàng).

```
def recommend_songs(song_list, spotify_data, n_songs=10):
    numerical_cols = song_cluster_pipeline.steps[0][1].feature_names_in_
    spotify_data_aligned = spotify_data[numerical_cols]

    # Tính vector trung tâm từ các bài hát đầu vào
    song_center = get_mean_vector(song_list, spotify_data)
    scaler = song_cluster_pipeline.steps[0][1]
    scaled_data = scaler.transform(spotify_data_aligned)
    song_center_scaled = scaler.transform(song_center.reshape(1, -1))

    # Tính khoảng cách và chọn các bài hát gần nhất
    distances = np.linalg.norm(scaled_data - song_center_scaled, axis=1)
    recommendations = spotify_data.iloc[np.argsort(distances)[:n_songs]]

    # Định dạng kết quả
    formatted_recommendations = []
    for _, row in recommendations.iterrows():
        song_entry = f"{row['track_name']} - {row['year']} - {row['artist_name']}"
        formatted_recommendations.append(song_entry)

    return formatted_recommendations
```

- Mục đích: Gợi ý các bài hát tương tự dựa trên danh sách các bài hát đầu vào.
- Cách hoạt động:
- Xác định các cột số (numerical_cols) từ pipeline.
- Lấy dữ liệu từ spotify_data dựa trên các cột số.

- Tính vector trung bình của các bài hát đầu vào bằng get_mean_vector.
- Chuẩn hóa dữ liệu và vector trung tâm.
- Tính khoảng cách Euclide giữa các bài hát trong spotify_data và vector trung tâm.
- Chọn n_songs bài hát gần nhất dựa trên khoảng cách.

- Cuối cùng ta in ra dựa trên lịch sử nghe nhạc.

```
song_list = [
    {'name': 'Returnelle', 'year': 2017},
    {'name': 'This Kind of Love', 'year': 2021},
    {'name': 'Reach Down', 'year': 2002}
]

# Gọi hàm recommend_songs
recommended_songs = recommend_songs(song_list, spotify_data=df)

# In kết quả
for song in recommended_songs:
    print(f"- '{song}'")
```

```
- 'The Way You Are - 2011 - Secrets Between Sailors'
- 'Little Maggie - 2011 - Snakefarm'
- 'Waiting for Me (feat. Stephen Kellogg & the Sixers) - 2013 - Taylor Carson'
- 'Let's Get Low Down - 2012 - Jon Cleary'
- 'Jurame - 2011 - Kevin Ceballo'
- 'Here Again (feat. The Combo Bárbaro) - 2012 - Quantic'
- 'Negro Rei - 2011 - Toquinho'
- '三十歲前要完成的事 - 2012 - Phil Lam'
- 'Why Are You Such A Bitch - 2011 - The Leonard Washingtons'
- 'Guys With Trucks - 2014 - Faux Paw'
```

5 Demo giao diện

Link: https://drive.google.com/drive/folders/1LTLV7Rpiu70pzy6EICcZeOrDkTbx_qQW

Link Google Colab: https://colab.research.google.com/drive/1LE8X8eQcwH_LPB9Kjj8lZdjX8RGqu5JM#scrollTo=6Xg4dDpIfUo1

6 Tài liệu tham khảo

1. MCSLearn-1 : MTH10605-24-25-1 : Resources
2. Report_Python_Sample.pdf
3. Simple EDA of spotify dataset
4. cmulay/spotify-eda: Spotify Exploratory Data Analysis
5. How Spotify's Algorithm Works? A Complete Guide to Spotify Recommendation System [2022] | Music Tomorrow Blog
6. Social recommendation using Euclidean embedding | IEEE Conference Publication | IEEE Xplore
7. Machine Learning pipeline — Machine Learning cho dữ liệu dạng bảng