ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC

HỆ THỐNG TƯ VẤN

Ngành: Khoa Học Dữ Liệu

Lớp: 20KDL1

Nhóm:

Giảng viên: Huỳnh Thanh Sơn

◇ THÀNH VIÊN ◇

STT	Tên	MSSV
1	Nguyễn Quốc Tiến	20280098
2	Dương Vi Doanh	20280018
3	Phù Chí Đạt	20280015
4	Võ Thái Bình	20280007

Ngày 27 tháng 12 năm 2023

Mục lục

⋆ BÁO CÁO ⋆

⋆ HỆ THỐNG ĐỀ XUẤT DANH SÁCH NHẠC ⋆

1 Lời nói đầu

Ngày nay, âm nhạc đã trở thành một phần không thể thiếu trong cuộc sống. Lợi ích của âm nhạc là vô cùng to lớn, nó không chỉ giúp bạn thư giãn mà còn có thể cải thiện chất lượng giấc ngủ, ngăn ngừa chứng trầm cảm, giảm đau, kiểm soát cơn thèm ăn để làm đẹp vóc dáng... Nếu không nghe nhạc mỗi ngày, bạn đã bỏ lỡ rất nhiều lợi ích tốt cho sức khỏe.

Nhạc có rất nhiều thể loại, và mọi người đều nghe nhạc với sở thích , nhu cầu, mục đích không giống nhau. Do đó, hệ thống đề xuất là công cụ lọc thông tin mạnh mẽ có thể giúp chúng ta tận dụng nguồn dữ liệu dồi dào hiện có để giúp cho việc đưa ra lựa chọn những bài nhạc phù hợp với nhu cầu, sở thích của mỗi cá nhân một cách dễ dàng hơn. Có hai lợi ích của việc sử dụng hệ thống đề xuất: Một mặt, nó có thể giảm lượng lớn công sức tìm kiếm sản phẩm của người dùng và giảm thiểu vấn đề quá tải thông tin. Mặt khác, nó có thể tăng giá trị kinh doanh cho các nhà cung cấp dịch vụ trực tuyến và trở thành nguồn doanh thu quan trọng.

2 Loading library and dataset - Đọc thư viện và bộ dữ liệu

Chú thích : Importing Libraries - Thêm những thư viện cần thiết

```
import os
import numpy as np
import pandas as pd

import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.pipeline import Pipeline
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist
from google.colab import drive
```

Chú thích :

- **Os :** Thư viện phục vụ cung cấp các hàm tương tác với hệ điều hành, giúp thực hiện các thao tác như đọc/ghi file, thực thi các lệnh hệ thống, và quản lý thư mục.
- **Pandas :** Thư viện phục vụ các thao tác xử lý, phân tích dữ liệu,...
- **Numpy :** Thư viện toán học hỗ trợ xử lí các dãy số, ma trận nhiều chiều,...
- **Seaborn :** Là một thư viện cung cấp các giao diện cao cấp cho việc vẽ đồ thị thống kê, giúp làm cho việc tạo biểu đồ trở nên đơn giản và mô phỏng đẹp mắt.
- **plotly.express :** Là một phần của thư viện Plotly, cung cấp một cách dễ sử dụng để tạo biểu đồ tương tác.
- **matplotlib.pyplot :** Là một thư viện trực quan hóa mạnh mẽ. pyplot là một module trong Matplotlib cung cấp các hàm để tạo các biểu đồ giống như MATLAB.
- **Sklearn :** là một thư viện máy học phổ biến. Cung cấp nhiều công cụ cho huấn luyện mô hình, chuẩn hóa dữ liệu, và đánh giá hiệu suất mô hình.
- **google.colab :** Là một gói cho phép bạn tương tác với Google Colab.

Reading dataset: đọc bộ dữ liệu

```
data = pd.read_csv("data.csv")
genre_data = pd.read_csv('data_by_genres.csv')
year_data = pd.read_csv('data_by_year.csv')
```

Nguồn bộ dữ liệu : Kaggle.
https://www.kaggle.com/code/vatsalmavani/music-recommendation-system-using-spotify-dataset

3 Data Understanding by EDA and Visualization - Hiểu dữ liệu bằng EDA và trực quan hóa

3.1 Exploratory Data Analysis - Phân tích dữ liệu tổng quan

Bộ dữ liệu được lấy từ Spotify, để đảm bảo rằng bạn có quyền truy cập và lấy dữ liệu. Bạn cần cài đặt Spotify, sau đó tạo một ứng dụng trên trang Spotify Developer và lưu lại Client ID và secret

key. Điều này là để xác thực ứng dụng của bạn khi truy cập Spotify API.
Cách lấy dữ liệu được trình bày cụ thể ở phần 5.1 Build Recommender System. **Chú thích :** Tập dữ liệu Spotily bao gồm 3 tệp :

- **data:** Chứa các đặt trưng âm nhạc như acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo, valence, và popularity, ngoài ra còn có các thông tin như nghệ sĩ, năm phát hành, ID của bài hát, key, mode, tên bài hát, và ngày phát hành,...
- **genres_data:** Ngoài chứa các đặt trưng âm nhạc, còn chứa thêm cột 'genres', cho biết thể loại của bài hát.
- **year_data:** Giữ lại những đặc trưng chính của âm nhạc, như năng lượng, tempo, valence, và các thông số quan trọng khác. Dữ liệu được chuẩn hóa để đảm bảo các dòng dữ liệu được thống nhất và thích hợp cho mô hình.

```
# Info of the 3 dataset
print(data.info())
print(genre_data.info())
print(year_data.info())
```

Kết quả : Bao gồm số lượng dòng và cột, số lượng giá trị không rỗng, kiểu dữ liệu của mỗi cột, và lượng bộ nhớ sử dụng.

Bộ dữ liệu	number of rows and columns	dtypes	memory usage
data	170653 rows và 19 columns	float64(9), int64(6), object(4)	24.7+ MB
genre_data	2973 rows và 14 columns	float64(11), int64(2), object(1)	325.3+ KB
year_data	100 rows và 14 columns	float64(11), int64(3)	11.1 KB

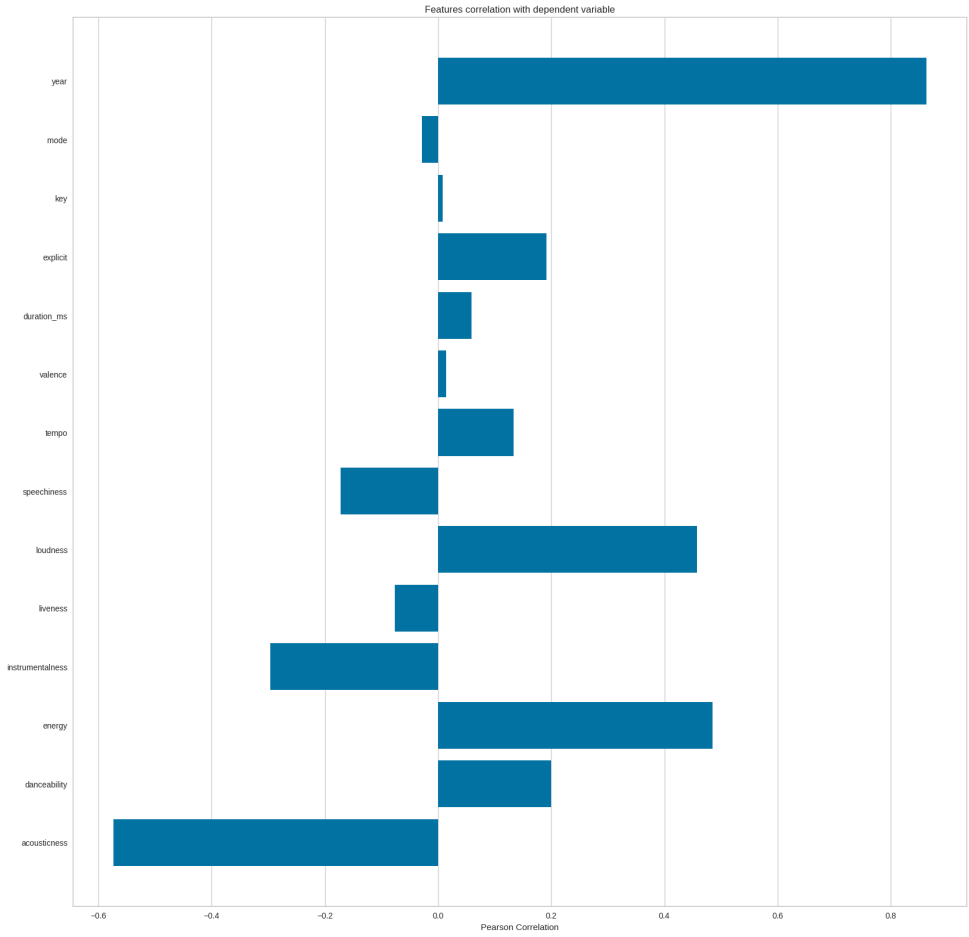
Từ các kết quả trên ta nhận thấy rằng bộ dữ liệu ta đang sử dụng không tồn tại dữ liệu không hợp lệ như 0 hoặc Null nên ta không cần tiến hàng xử lý dữ liệu.

3.2 Data Visualization - Trực quan hóa dữ liệu

```
from yellowbrick.target import FeatureCorrelation
feature_names = ['acousticness', 'danceability', 'energy', 'instrumentalness',
      'liveness', 'loudness', 'speechiness', 'tempo',
          'valence','duration_ms','explicit','key','mode','year']
X, y = data[feature_names], data['popularity']

# Create a list of the feature names
features = np.array(feature_names)
# Instantiate the visualizer
visualizer = FeatureCorrelation(labels=features)
```

```
plt.rcParams['figure.figsize']=(20,20)
visualizer.fit(X, y)   # Fit the data to the visualizer
visualizer.show()
```

Kết quả:



Hệ số tương quan là một độ đo thống kê mô tả mức độ mối quan hệ tuyến tính giữa hai biến. Hệ số tương quan Pearson, thường được ký hiệu là "r," là một trong những phổ biến nhất và phổ quát nhất. Hệ số này có giá trị từ -1 đến 1.

Giá trị 1: Tương quan Hoàn toàn Thuận (Perfect Positive Correlation)

- Khi r = 1, có mối quan hệ tuyến tính hoàn toàn thuận giữa hai biến. Điều này có nghĩa là khi giá trị của biến thứ nhất tăng, giá trị của biến thứ hai cũng tăng theo cùng một tỷ lệ cố định, và ngược lại.

Giá trị -1: Tương quan Hoàn toàn Nghịch (Perfect Negative Correlation)

- Khi r = -1, có mối quan hệ tuyến tính hoàn toàn nghịch giữa hai biến. Điều này có nghĩa là khi giá trị của biến thứ nhất tăng, giá trị của biến thứ hai giảm theo cùng một tỷ lệ cố định, và ngược lại.

Giá trị 0: Không có Tương quan Tuyến tính

- Khi r = 0, không có mối quan hệ tuyến tính giữa hai biến. Tăng hoặc giảm của một biến không diễn giải sự thay đổi của biến kia.

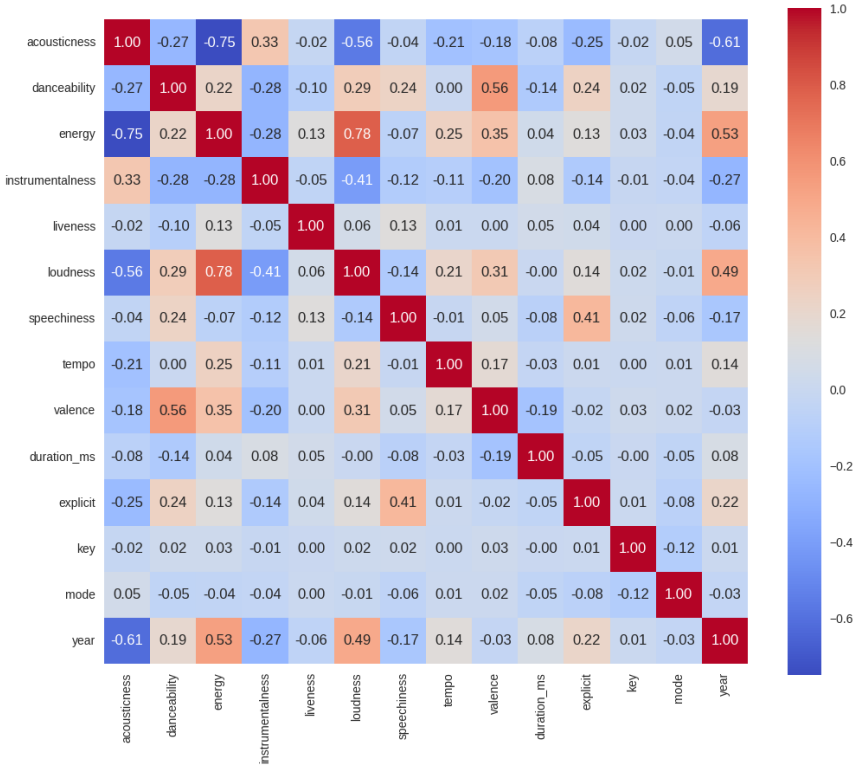
Giá trị ở Giữa -1 và 1: Tương quan Tuyến tính Mạnh hoặc Yếu

- Giá trị gần -1 hoặc 1 cho thấy một mối quan hệ tuyến tính mạnh giữa hai biến. Giá trị gần 0 hơn thì mối quan hệ tuyến tính yếu hơn.

Trong ngữ cảnh của âm nhạc, feature "year" là năm phát hành của một bản nhạc, mối quan hệ mạnh thuận có thể chỉ ra rằng các bản nhạc được phát hành sau có xu hướng có độ phổ biến cao hơn. Điều này có thể phản ánh xu hướng người nghe có sự quan tâm cao đối với âm nhạc hiện đại hoặc các yếu tố khác như chiến lược quảng cáo, v.v.

```
# List of feature names
feature_names = ['acousticness', 'danceability', 'energy', 'instrumentalness',
       'liveness', 'loudness', 'speechiness', 'tempo',
           'valence','duration_ms','explicit','key','mode','year']

# Extract features
X = data[feature_names]
# Compute the correlation matrix
correlation_matrix = X.corr()

# Set the size of the plot
plt.rcParams['figure.figsize'] = (12, 10)
# Create a heatmap of the correlation matrix
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', square=True)
# Show the plot
plt.show()
```

Kết quả:



Ý nghĩa của Biểu đồ Heatmap:

- Heatmap giúp hiển thị mức độ tương quan giữa các đặc trưng trong một tập dữ liệu.
- Các ô có màu sáng hơn thường biểu thị tương quan mạnh hơn, trong khi các ô có màu tối hơn thường biểu thị tương quan yếu hơn hoặc không tương quan.
- Giá trị tương quan nằm trong khoảng từ -1 đến 1, với 1 là tương quan hoàn toàn thuận, -1 là tương quan hoàn toàn nghịch, và 0 là không có tương quan tuyến tính.

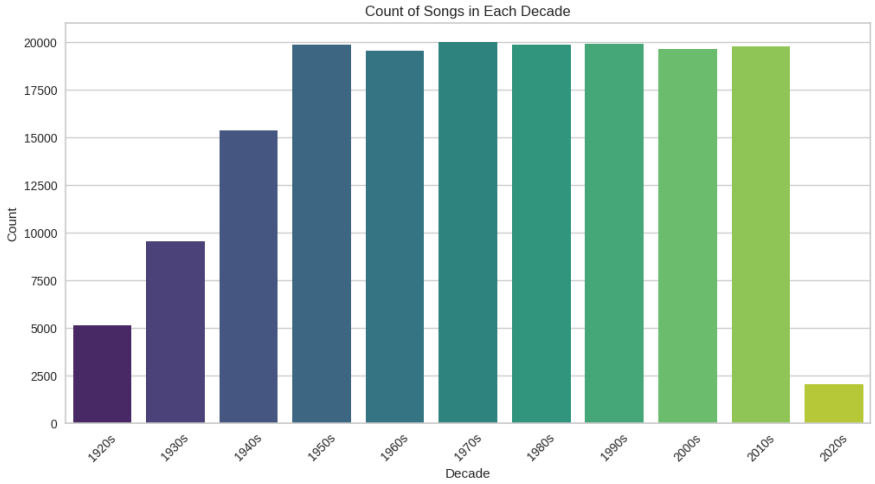
Music Over Time

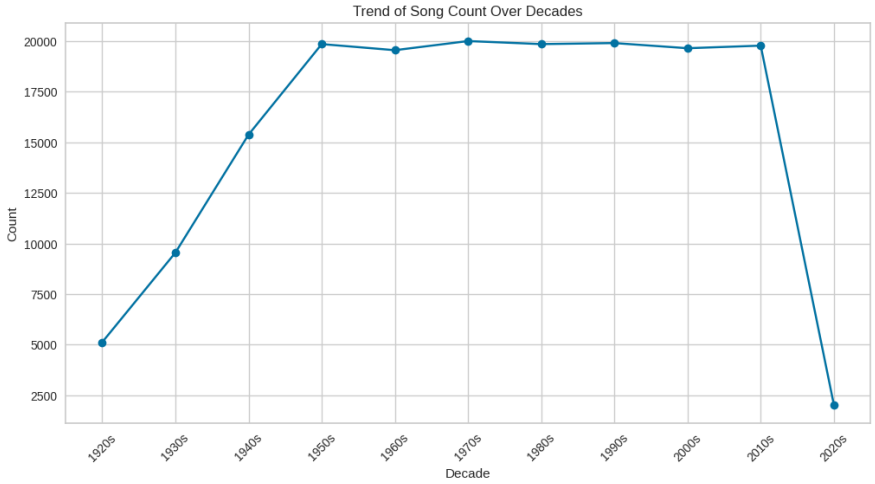
```
def get_decade(year):
    period_start = int(year/10) * 10
    decade = '{}s'.format(period_start)
    return decade

data['decade'] = data['year'].apply(get_decade)
```

```
# Set the size of the plot
plt.rcParams['figure.figsize'] = (12, 6)

# Create a bar plot to visualize the count of songs in each decade
```

```
sns.countplot(x='decade', data=data, palette='viridis')
plt.title('Count of Songs in Each Decade')
plt.xlabel('Decade')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()

# Create a line plot to visualize the trend of song count over decades
decade_counts = data['decade'].value_counts().sort_index()
plt.plot(decade_counts.index, decade_counts.values, marker='o', linestyle='-')
plt.title('Trend of Song Count Over Decades')
plt.xlabel('Decade')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

Kết quả:

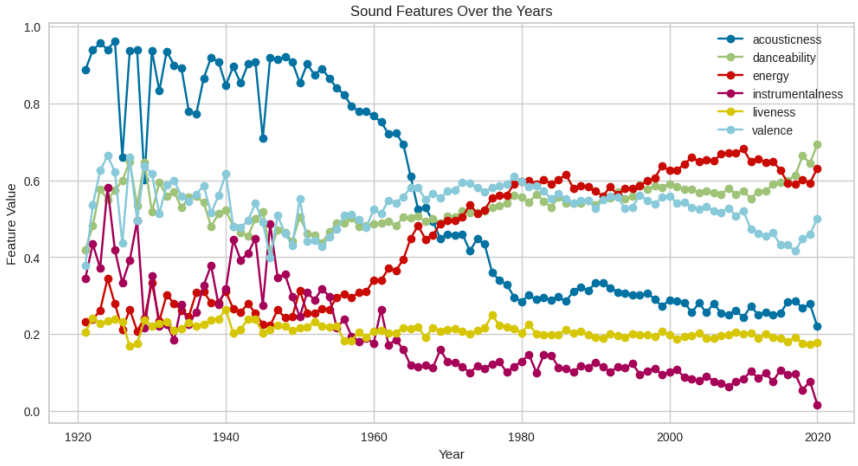




Biểu đồ cột: Hiển thị số lượng bài hát trong mỗi thập kỷ, giúp bạn thấy rõ sự phân phối và tần suất của các năm trong dữ liệu. Tạo biểu đồ cột (count plot) sử dụng Seaborn để hiển thị số lượng bài hát trong mỗi thập kỷ, kéo dài từ năm 1920 đến năm 2020. Biểu đồ cho thấy rằng thập kỉ 1950s đến 2010s thì số lượng bài hát được phát hành nhiều hơn so với các thập kỉ khác.
Biểu đồ đường: Cho thấy xu hướng tăng giảm của số lượng bài hát qua các thập kỷ, giúp đánh giá sự biến động và phát triển của âm nhạc qua thời gian.

```
sound_features = ['acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness',
    'valence']

# Set the size of the plot
plt.rcParams['figure.figsize'] = (12, 6)

# Create a line plot for each sound feature
for feature in sound_features:
    plt.plot(year_data['year'], year_data[feature], label=feature, marker='o',
        linestyle='-')

# Add labels and title
plt.title('Sound Features Over the Years')
plt.xlabel('Year')
plt.ylabel('Feature Value')
plt.legend() # Add a legend to identify each line
plt.show()
```

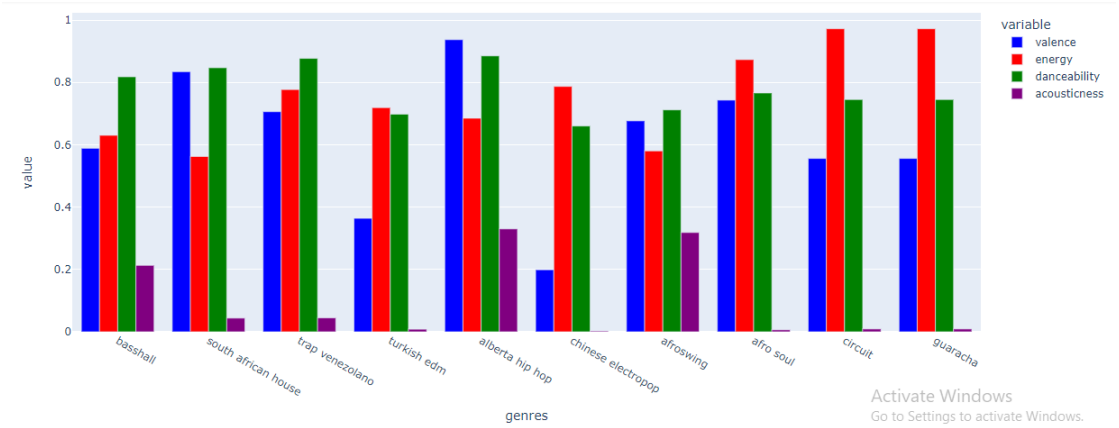
Kết quả:



Characteristics of Different Genres

```
top10_genres = genre_data.nlargest(10, 'popularity')

# Define custom colors for each feature
colors = {'valence': 'blue', 'energy': 'red', 'danceability': 'green', 'acousticness':
    'purple'}

# Create a bar chart with custom colors
fig = px.bar(
    top10_genres,
    x='genres',
    y=['valence', 'energy', 'danceability', 'acousticness'],
    barmode='group',
    color_discrete_map=colors
)
# Show the plot
fig.show()
```

Kết quả:



Biểu đồ cột hiển thị giá trị của các đặc trưng cho top 10 thể loại âm nhạc phổ biến nhất, với mỗi đặc trưng có một màu sắc tùy chỉnh nhất định.

4 Data Pre-processing - Tiền xử lý dữ liệu

Ta tiến hành kiểm tra xem mỗi giá trị có phải là giá trị thiếu hay không, tính tổng số lượng giá trị thiếu cho mỗi cột, và tiến hành trả về Series chứa số lượng giá trị thiếu cho mỗi cột.

```
def count_null_values(data):
    # Check if each value is a missing value
    null_values = data.isnull()
    # Calculate the total number of missing values for each column
```

```
    null_counts = null_values.sum()
    # Return a Series containing the total number of missing values for each column
    return null_counts
# Use Function
null_counts = count_null_values(data)
print("Number of missing values for each column:")
print(null_counts)
```

Kết quả :

- Bộ thứ nhất:

```
    Calculate the total number of missing values for each column
valence           0
year              0
acousticness      0
artists           0
danceability      0
duration_ms       0
energy            0
explicit          0
id                0
instrumentalness  0
key               0
liveness          0
loudness          0
mode              0
name              0
popularity        0
release_date      0
speechiness       0
tempo             0
dtype: int64
```

- Bộ thứ hai:

```
mode              0
genres            0
acousticness      0
danceability      0
duration_ms       0
energy            0
instrumentalness  0
```

```
liveness          0
loudness          0
speechiness       0
tempo             0
valence           0
popularity        0
key               0
dtype: int64
```

- Bộ thứ ba:

```
mode              0
year              0
acousticness      0
danceability      0
duration_ms       0
energy            0
instrumentalness  0
liveness          0
loudness          0
speechiness       0
tempo             0
valence           0
popularity        0
key               0
dtype: int64
```

Từ đây, ta thấy bộ dataframe ban đầu đã làm sạch và không cần tiến hành xử lí. Vì vậy, ta chuyển sang bước tiếp theo.

5 Build Recommender System - Xây dựng hệ thống đề xuất

Hệ thống đề xuất là hệ thống được thiết kế để giới thiệu mọi thứ cho người dùng dựa trên nhiều yếu tố khác nhau. Các hệ thống này dự đoán sản phẩm có khả năng nhất mà người dùng có nhiều khả năng mua và quan tâm nhất. Các công ty như Netflix , Amazon, Spotify v.v. sử dụng hệ thống đề xuất để giúp người dùng của họ xác định đúng sản phẩm cho họ. Các hệ thống đề xuất dựa trên mức độ phổ biến dựa trên xếp hạng các mặt hàng của tất cả người dùng. Hệ thống đề xuất dựa trên mức độ phổ biến hoạt động theo xu hướng. Về cơ bản, nó sử dụng các mặt hàng đang là xu hướng hiện nay.

5.1 K-means clustering Recommender System

K-mean clustering: K-means hoạt động thông qua một quá trình lặp để chia tập dữ liệu thành K cụm, trong đó mỗi điểm dữ liệu thuộc về cụm có trung bình (trọng tâm) gần nhất. Thuật toán phân cụm được áp dụng để phân cụm theo bài hát hoặc theo thể loại. Sau khi phân cụm, các kỹ thuật trực quan hóa như t-Distributed Stochastic Neighbor Embedding (t-SNE) đã được sử dụng để giảm không gian đặc trưng cao chiều xuống không gian 2D hoặc 3D để dễ dàng quan sát. Điều này giúp kiểm tra cách các bài hát được nhóm và mức độ gần nhau trong các cụm.

Clustering genres with K-means

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

cluster_pipeline = Pipeline([('scaler', StandardScaler()), ('kmeans',
    KMeans(n_clusters=10))])
X = genre_data.select_dtypes(np.number)
cluster_pipeline.fit(X)
genre_data['cluster'] = cluster_pipeline.predict(X)
```

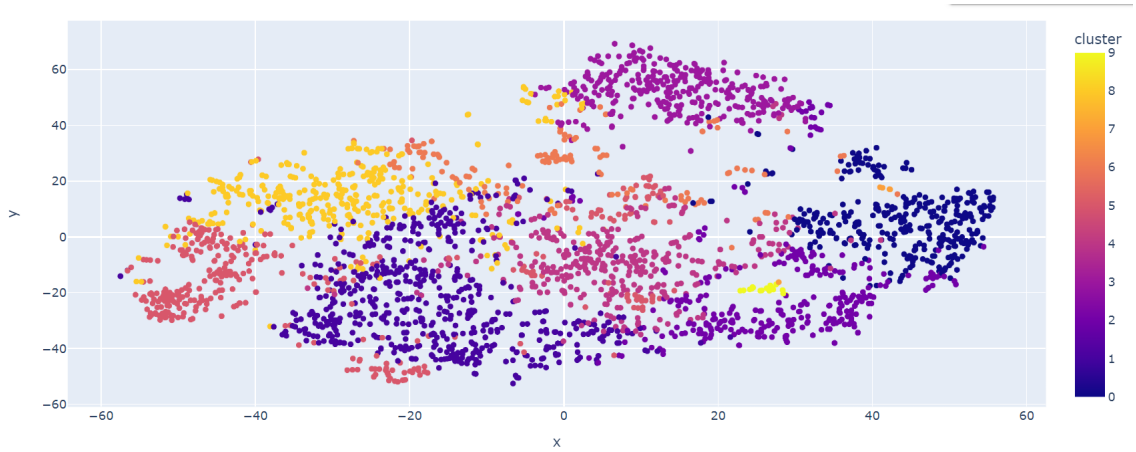
Các bước thực hiện

- Sử dụng Pipeline để xây dựng một chuỗi xử lý dữ liệu. Hai bước trong chuỗi là chuẩn hóa dữ liệu (StandardScaler) và phân cụm (KMeans với 10 cụm)
- Sau đó, họn các cột có kiểu dữ liệu số từ DataFrame genre_data. Các thuộc tính số này sẽ được sử dụng để xây dựng mô hình phân cụm.
- Áp dụng chuỗi xử lý dữ liệu và huấn luyện mô hình K-Means trên dữ liệu số. Dùng mô hình đã huấn luyện để dự đoán nhãn cụm cho mỗi điểm dữ liệu và gán kết quả vào cột 'cluster' trong DataFrame genre_data.
- Kết quả là một DataFrame mới (genre_data) chứa thông tin về nhãn cụm của mỗi điểm dữ liệu dựa trên mô hình K-Means. Cột 'cluster' giờ đây là một phần quan trọng của dữ liệu, dựa vào 'cluster' có thể biết dữ liệu thuộc cụm nào.

```
from sklearn.manifold import TSNE

tsne_pipeline = Pipeline([('scaler', StandardScaler()), ('tsne', TSNE(n_components=2,
    verbose=1))])
genre_embedding = tsne_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=['x', 'y'], data=genre_embedding)
```

```
projection['genres'] = genre_data['genres']
projection['cluster'] = genre_data['cluster']

fig = px.scatter(
    projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'genres'])
fig.show()
```

Kết quả:



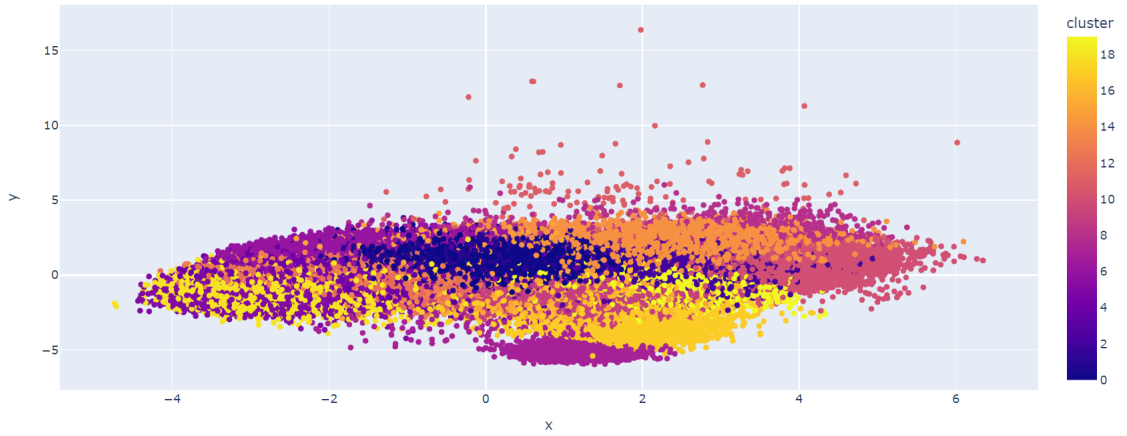
Kết quả là một biểu đồ scatter 2D, màu sắc thể hiện nhãn cụm của điểm đó, và thông tin thêm khi di chuột qua. Biểu đồ này giúp hiểu rõ hơn về phân bố và cấu trúc của các nhóm cụm trong dữ liệu.

Clustering songs with K-means

```
from sklearn.decomposition import PCA

pca_pipeline = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n_components=2))])
song_embedding = pca_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=['x', 'y'], data=song_embedding)
projection['title'] = data['name']
projection['cluster'] = data['cluster_label']

fig = px.scatter(
    projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'title'])
fig.show()

song_cluster_pipeline = Pipeline([('scaler', StandardScaler()),
                                  ('kmeans', KMeans(n_clusters=20,
                                   verbose=False))
                                 ], verbose=False)
```

```
X = data.select_dtypes(np.number)
number_cols = list(X.columns)
song_cluster_pipeline.fit(X)
song_cluster_labels = song_cluster_pipeline.predict(X)
data['cluster_label'] = song_cluster_labels
```

Kết quả:



Thực hiện giảm chiều dữ liệu từ không gian nhiều chiều xuống không gian 2 chiều bằng phương pháp Principal Component Analysis (PCA), sau đó vẽ biểu đồ scatter để hiển thị dữ liệu giảm chiều và màu sắc biểu thị nhãn cụm của mỗi điểm dữ liệu. Kết quả là một biểu đồ scatter 2D, màu sắc thể hiện nhãn cụm của điểm đó, và thông tin thêm khi di chuột qua. Biểu đồ này giúp hiểu rõ hơn về phân bố và cấu trúc của các nhóm cụm trong dữ liệu, đặc biệt là sau khi áp dụng giảm chiều dữ liệu PCA.

5.2 Build Recommender System using spotily api to get data

Các bước để thực hiện gọi API bằng cách truy xuất dữ liệu của các nghệ sĩ:

- Đầu tiên cần thiết lập tài khoản của bạn: Đăng nhập vào Spotify Developer Dashboard
- Sau đó, tạo một project mới:
  - Để tạo ứng dụng, vào Trang quản lý của bạn, nhấp vào nút Tạo một ứng dụng và nhập thông tin sau:
    - Tên project. Ví dụ: My new project
    - Mô tả project. Ví dụ: This is my first Sportify project
- Tiếp theo, cần yêu cầu một token truy cập:
  - Mã truy cập là một chuỗi chứa thông tin đăng nhập và quyền truy cập có thể được sử dụng để truy cập một tài nguyên cụ thể (ví dụ: nghệ sĩ, album hoặc bản nhạc) hoặc dữ liệu người dùng (ví dụ: hồ sơ của bạn hoặc danh sách phát của bạn)

- Để yêu cầu mã truy cập, bạn cần lấy Client_ID và Client Secret của mình:
  - Truy cập Trang quản lý của bạn.
  - Nhấp vào tên project bạn vừa tạo (My new project).
  - Nhấp vào nút Cài đặt.
  - Client ID nằm ở đây. Client Secret có thể được tìm thấy sau View client secret link.
- Với thông tin xác thực trong tay, bạn đã sẵn sàng yêu cầu mã truy cập. Hướng dẫn này sử dụng Client Credentials, vì vậy chúng ta cần:
  - Gửi một yêu cầu POST đến URI điểm cuối mã thông báo.
  - Thêm tiêu đề Content-Type với giá trị application/x-www-form-urlencoded.
  - Thêm một phần thân HTTP chứa Client ID và Client Secret, cùng với tham số grant_type được đặt thành client_credentials.

```
curl -X POST "https://accounts.spotify.com/api/token" \
     -H "Content-Type: application/x-www-form-urlencoded" \
     -d "grant_type=client_credentials&client_id=your-client-id&client_secret=your-client-secret"
```

- Phản hồi sẽ trả về một mã truy cập có hiệu lực trong 1 giờ.

```
{
  "access_token": "BQDBKJ5eo5jxbtpWjVOj7ryS84khybFpP_lTqzV7uV-T_m0cTfwvdn5BnBSKPxKgEb11",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

- Cuối cùng, yêu cầu dữ liệu nghệ sĩ: Cách dễ dàng để lấy Spotify ID của một nghệ sĩ là sử dụng ứng dụng Spotify trên máy tính:
  - Tìm kiếm nghệ sĩ: Sử dụng chức năng tìm kiếm trong ứng dụng Spotify để tìm kiếm nghệ sĩ mà bạn quan tâm.
  - Nhấp vào biểu tượng ba chấm từ hồ sơ nghệ sĩ: Sau khi bạn đã tìm thấy nghệ sĩ, hãy nhấp vào biểu tượng ba chấm ở góc phải của hồ sơ nghệ sĩ. Chọn Share > Copy link to artist.
  - Trong menu xuất hiện, chọn "Share" và sau đó chọn "Copy link to artist". Điều này sẽ sao chép đường liên kết của nghệ sĩ vào clipboard của bạn. Spotify ID là phần sau open.spotify.com/artist: Dán đường liên kết sao chép vào một nơi nào đó, và Spotify ID sẽ là giá trị ngay sau open.spotify.com/artist. Ví dụ, nếu đường liên kết là https://open.spotify.com/artist/4Z8W4fKeB thì Spotify ID là 4Z8W4fKeB5YxbusRsdQVPb.
- Nếu mọi thứ diễn ra suôn sẻ, API sẽ trả về phản hồi JSON:

```
{
  "external_urls": {
    "spotify": "https://open.spotify.com/artist/4Z8W4fKeB5YxbusRsdQVPb"
  },
  "followers": {
    "href": null,
    "total": 7625607
  },
  "genres": [
    "alternative rock",
    "art rock",
    "melancholia",
    "oxford indie",
    "permanent wave",
    "rock"
  ],
  "href": "https://api.spotify.com/v1/artists/4Z8W4fKeB5YxbusRsdQVPb",
  "id": "4Z8W4fKeB5YxbusRsdQVPb",
  "images": [
    {
      "height": 640,
      "url": "https://i.scdn.co/image/ab6761610000e5eba03696716c9ee605006047fd",
      "width": 640
    },
    {
      "height": 320,
      "url": "https://i.scdn.co/image/ab67616100005174a03696716c9ee605006047fd",
      "width": 320
    },
    {
      "height": 160,
      "url": "https://i.scdn.co/image/ab6761610000f178a03696716c9ee605006047fd",
      "width": 160
    }
  ],
  "name": "Radiohead",
  "popularity": 79,
  "type": "artist",
  "uri": "spotify:artist:4Z8W4fKeB5YxbusRsdQVPb"
}
```


Giải thích các hàm áp dụng:

```
def find_song(name, year):
    song_data = defaultdict()
    results = sp.search(q= 'track: {} year: {}'.format(name,year), limit=1)
    if results['tracks']['items'] == []:
        return None

    results = results['tracks']['items'][0]
    track_id = results['id']
    audio_features = sp.audio_features(track_id)[0]

    song_data['name'] = [name]
    song_data['year'] = [year]
    song_data['explicit'] = [int(results['explicit'])]
    song_data['duration_ms'] = [results['duration_ms']]
    song_data['popularity'] = [results['popularity']]

    for key, value in audio_features.items():
        song_data[key] = value
```

```
    return pd.DataFrame(song_data)
```

Chức năng của hàm Find_song:

• Nó lấy tên và năm của bài hát làm thông số đầu vào.

• Sử dụng phương pháp tìm kiếm của Spotipy, nó tìm kiếm bản nhạc khớp với tên và năm được cung cấp

• Nếu tìm thấy một bản nhạc, nó sẽ thu thập nhiều tính năng khác nhau như thời lượng, mức độ phổ biến và tính năng âm thanh bằng cách sử dụng ID của bản nhạc thu được từ tìm kiếm

• Cuối cùng, nó trả về DataFrame chứa dữ liệu bài hát đã thu thập.

```
def get_song_data(song, spotify_data):

    try:
        song_data = spotify_data[(spotify_data['name'] == song['name'])
                                 & (spotify_data['year'] == song['year'])].iloc[0]
        return song_data

    except IndexError:
        return find_song(song['name'], song['year'])


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[number_cols].values
        song_vectors.append(song_vector)

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


def flatten_dict_list(dict_list):
```

```
    flattened_dict = defaultdict()
    for key in dict_list[0].keys():
        flattened_dict[key] = []

    for dictionary in dict_list:
        for key, value in dictionary.items():
            flattened_dict[key].append(value)

    return flattened_dict


def recommend_songs( song_list, spotify_data, n_songs=10):

    metadata_cols = ['name', 'year', 'artists']
    song_dict = flatten_dict_list(song_list)

    song_center = get_mean_vector(song_list, spotify_data)
    scaler = song_cluster_pipeline.steps[0][1]
    scaled_data = scaler.transform(spotify_data[number_cols])
    scaled_song_center = scaler.transform(song_center.reshape(1, -1))
    distances = cdist(scaled_song_center, scaled_data, 'cosine')
    index = list(np.argsort(distances)[:, :n_songs][0])

    rec_songs = spotify_data.iloc[index]
    rec_songs = rec_songs[~rec_songs['name'].isin(song_dict['name'])]
    return rec_songs[metadata_cols].to_dict(orient='records')
```

Chức năng của các hàm còn lại:

• get_song_data(song, Spotify_data): Truy xuất dữ liệu bài hát từ tập dữ liệu được cung cấp (spotify_data) hoặc bằng cách truy vấn Spotify bằng hàm find_song.

• get_mean_vector(song_list, Spotify_data): Tính toán vectơ trung bình của các đặc điểm số cho danh sách các bài hát.

• Flatten_dict_list(dict_list): Làm phẳng danh sách từ điển thành một từ điển duy nhất.

• recommend_songs(song_list, Spotify_data, n_songs=10): Đề xuất bài hát dựa trên danh sách bài hát được cung cấp. Nó tính toán khoảng cách cosine giữa vectơ trung bình của các bài hát đầu vào và tập dữ liệu Spotify để đề xuất các bài hát tương tự.

```
recommend_songs([{'name': 'City of stars', 'year': 1993},
                 {'name': 'Can\'t take my eyes off you', 'year': 2017},
                 {'name': 'Can\'t help falling in love', 'year': 1961},
```

```
            {'name': 'Take me to your heart', 'year':1999},
            {'name': 'Suspicious Minds', 'year': 1969},
            {'name': 'Both sides now', 'year': 2021}], data)
```

Kết quả:

```
[{'name': 'you were good to me',
  'year': 2019,
  'artists': "['Jeremy Zucker', 'Chelsea Cutler']"},
 {'name': 'Happiest Year', 'year': 2019, 'artists': "['Jaymes Young']"},
 {'name': 'Dancing On My Own', 'year': 2018, 'artists': "['Calum Scott']"},
 {'name': 'Hey There Delilah',
  'year': 2005,
  'artists': '["Plain White T\'s"]'},
 {'name': 'Whiskey And You', 'year': 2015, 'artists': "['Chris Stapleton']"},
 {'name': 'Big Black Car', 'year': 2009, 'artists': "['Gregory Alan Isakov']"},
 {'name': "Where's My Love - Acoustic", 'year': 2017, 'artists': "['SYML']"},
 {'name': 'Harvest Moon', 'year': 1992, 'artists': "['Neil Young']"},
 {'name': "Heart's Content", 'year': 2012, 'artists': "['Brandi Carlile']"},
 {'name': 'She', 'year': 2019, 'artists': "['dodie']"}]
```

6 Kết luận

Hệ thống đề xuất là một công nghệ mới mạnh mẽ để trích xuất giá trị bổ sung cho doanh nghiệp từ cơ sở dữ liệu người dùng của nó. Hệ thống mang lại lợi ích cho người dùng bằng cách cho phép họ tìm thấy các nhiều bài nhạc mà họ thích một cách dễ dàng hơn. Ngược lại, họ giúp doanh nghiệp bằng cách tạo ra nhiều doanh thu hơn.

Về phần báo cáo mà nhóm em đã thực hiện. Từ một bộ dữ liệu chứa thông tin về của hơn 271000 bài nhạc, và các người dùng đã đánh giá nó, nhóm đã trực quan hóa dữ liệu, xử lý các dữ liệu bất hợp lý, cũng như đưa ra các đề xuất cho người dùng theo nhiều cách khác nhau. Ở đây, nếu người dùng của chúng ta là người dùng mới, chúng ta sẽ đề xuất cho người dùng dựa trên top 10 các cuốn sách nổi tiếng nhất. Nếu là người dùng đã từng sử dụng nền tảng trực tuyến của chúng ta, khi họ nhập vào ID mình thì có thể thấy được top 5 cuốn sách yêu thích nhất của bản thân, bên cạnh đó hệ thống cũng sẽ đề xuất cho họ những cuốn sách mà họ có thể thích. Ngoài ra, người dùng cũng có thể tìm kiếm sự đề xuất bằng cách nhập vào tên của cuốn sách , hệ thống sẽ tiến hành đề xuất dựa trên nội dung, hay dựa trên sự tương đồng của sản phẩm. Bằng các cách lọc này hệ thống sẽ đưa ra cho người dùng các cuốn sách tương tự cuốn sách mà họ tìm kiếm - nếu cuốn sách mà họ tìm kiếm nằm trong bộ dữ liệu và có một độ phổ biến nhất định, ngược lại những cuốn sách chưa phổ biến thì sẽ không có sự đề xuất nào cả.

Nhìn chung về bài làm mà nhóm đã thực hiện code vẫn chưa thật sự tối ưu. Hơn nữa, hệ thống của chúng ta vẫn còn nhiều thiếu sót, chẳng hạn khi ta có một cuốn sách mới và cuốn sách này chưa phổ biến nhưng có đánh giá khá tốt, nghĩa là cuốn sách này sẽ có xu hướng có thể nổi tiếng trong thời gian tới, đối với cách làm mà nhóm thực hiện vẫn chưa cải thiện được trường hợp này. Nhóm em sẽ cố gắng cải thiện trong thời gian tới để hệ thống được hoàn thiện hơn.

Cuối cùng, âm nhạc không chỉ giúp cho chúng ta thư giản, mà còn mang lại nhiều lợi ích hơn thế .Nghe nhạc giúp ta cải thiện sự tập trung và tăng cường khả năng tư duy, phân tích ; vốn từ của chúng ta cũng sẽ được mở rộng thông qua việc nghe nhạc .Âm nhạc còn là một hình thức giải trí , giảm căng thẳng ... Mọi người nên khắc sâu thói quen nghe nhạc mỗi ngày vì những lợi ích đáng quý của nó.

Hy vọng bằng cách sử dụng hệ thống đề xuất nhạc, chúng ta có thể tìm kiếm những sản phẩm âm nhạc phù hợp với sở thích , nhu cầu dễ dàng hơn, từ đó mà chúng ta có thể nuôi dưỡng thói quen nghe nhạc mỗi ngày.

7 Tài liệu tham khảo

https://www.kaggle.com/code/vatsalmavani/music-recommendation-system-using-spotify-dataset

https://hellobacsi.com/tam-ly-tam-than/tram-cam-roi-loan-cam-xuc/loi-ich-cua-viec-nghe-nhac/