Final Project Presentation

Fall 2024 Data Science Bootcamp
Topic: Music Recommender System

Spotify

Group 2 Members:
Thomas Zhang, Martin Chen, Kehan Lin

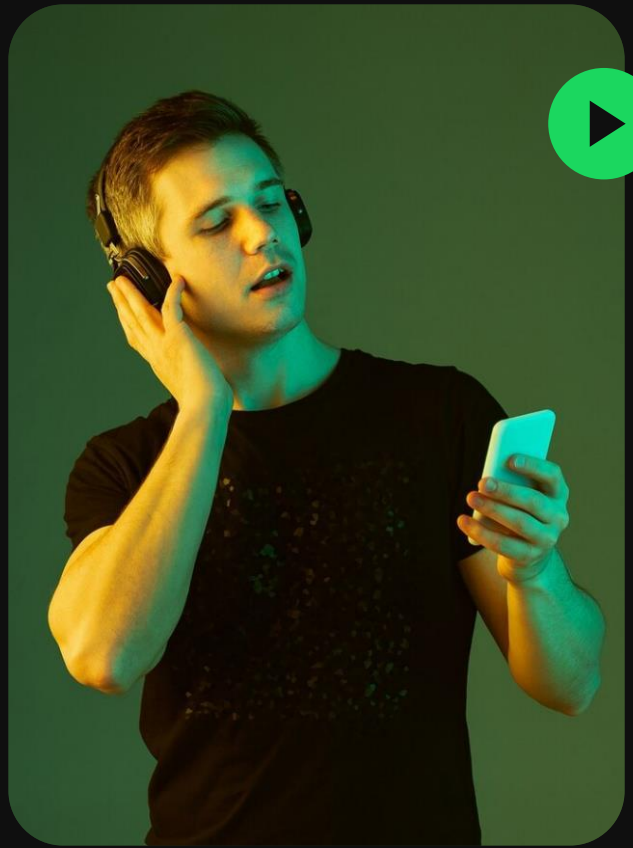


Table of Contents

#	Topics		
1	EDA	MT	
2	Data Visualization	MT	
3	Content-based Filtering	FIN	
4	Collaborative Filtering	FIN	
5	Spotipy	FIN	
6	Next Steps	FIN	

Content-based Filtering

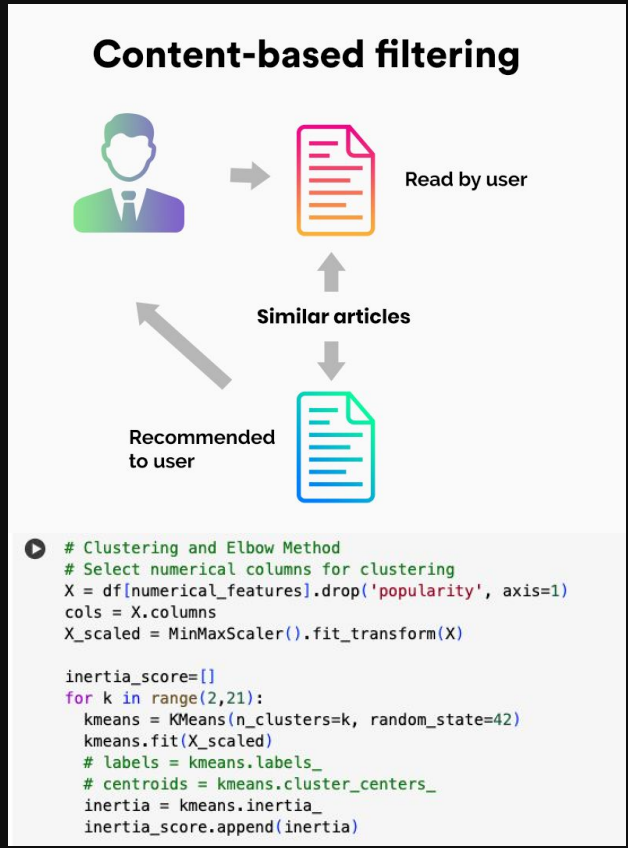
Content-based Filtering relies on the features of the items (e.g., acousticness, energy) when making suggestions to users based on what items they have previously liked (the songs they search).

K-means Clustering is used to group data points into clusters by using a mathematical distance measure

Method 1: K-means + kNN

- K-means Clustering partitions tracks (data points)
- KNN find the k nearest songs by Euclidean distance (in the same cluster) to recommend

Preprocessing: Normalizing (using min-max scaler) all non-target features




```
# Clustering and Elbow Method
# Select numerical columns for clustering
X = df[numerical_features].drop('popularity', axis=1)
cols = X.columns
X_scaled = MinMaxScaler().fit_transform(X)

inertia_score=[]
for k in range(2,21):
  kmeans = KMeans(n_clusters=k, random_state=42)
  kmeans.fit(X_scaled)
  # labels = kmeans.labels_
  # centroids = kmeans.cluster_centers_
  inertia = kmeans.inertia_
  inertia_score.append(inertia)
```

Content-based Filtering

K-means Clustering

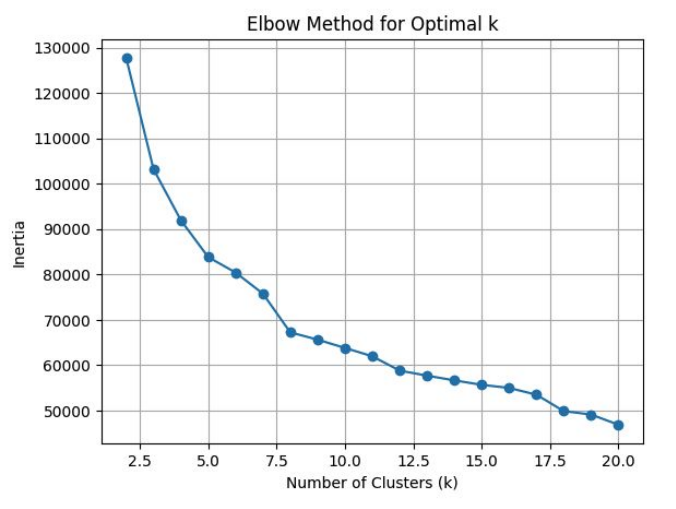
(Challenge) Choose the right K

Evaluation: Attempt to use Elbow Method to find the prime K

Issue: K always decreasing (range 2 to 51, 101, 151…)

Solution: Choose the optimal K=20 when range (2, 21)

- Other potential clustering methods



Content-based Filtering

k-nearest Neighbors (kNN)

A supervised learning classifier which uses proximity to make classifications or predictions about the grouping of an individual data point

Intuitive because we need to recommend k songs for each user's search

Steps:

1. Find the song idx in data.csv and its corresponding cluster
2. Train kNN model on that cluster and find the k nearest songs for recommendation

```python
# Function to recommend K nearest songs
def recommend_songs(song_id, K=5):

    song_index = df[df['id']==song_id].index[0]

    # Get the cluster of the input song
    song_cluster = df.loc[song_index, 'cluster']

    # Filter songs in the same cluster
    cluster_data = df[df['cluster'] == song_cluster]

    # Get features of the cluster
    cluster_features = X_scaled[df['cluster'] == song_cluster]

    # Train KNN on the cluster
    knn = NearestNeighbors(n_neighbors= K + 1, metric='euclidean')
    knn.fit(cluster_features)

    # Find the index of the input song within the cluster
    input_song_features = X_scaled[song_index].reshape(1, -1)
    distances, indices = knn.kneighbors(input_song_features)

    # Remove the input song from the results and get the recommended song indices
    recommended_indices = indices[0][1:]

    # Get the song details for the recommendations
    recommendations = cluster_data.iloc[recommended_indices]
    return recommendations
```

```
[71]: recommend_songs('65G7XDGcybJiGywSCXJiL5') # Animals by Martin Garrix
```

[71]:

	valence	year	acousticness	artists	danceability	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	name	popularity	release_date	speechiness
154130	0.1930	2013	0.000909	['Taku Iwasaki']	0.576	0.822	0	3NbcxD3uC4aDsv87lwK1WM	0.537	1	0.1100	-8.456	1	Overdrive	0.53	2013-03-29	0.110
137700	0.2570	2004	0.000912	['The Prodigy']	0.563	0.924	0	1y6MO9mVEUluMcI4RaIlGR	0.578	2	0.0351	-2.703	1	Spitfire	0.38	2004-08-23	0.041
105203	0.2320	2002	0.002720	['Linkin Park', 'Jonathan Davis']	0.471	0.927	0	6bz9irEuD1lus2hcHlKRZh	0.537	1	0.1150	-5.760	1	1Stp Klosr (The Humble Brothers Reanimation) [...	0.39	2002-07-29	0.049
89390	0.1410	2006	0.000398	['Breaking Benjamin']	0.468	0.774	0	5MNxNuo0XSHx7MPXbsR57W	0.624	0	0.1470	-4.769	1	You	0.48	2006-01-01	0.038
89514	0.0697	2006	0.000114	['Lacuna Coil']	0.506	0.724	0	6vS5siwSidltcWFXskcpAA	0.628	0	0.1510	-7.086	1	Our Truth	0.52	2006	0.028

```
recommend_songs('2uu2aGqA2UblCg581Q7l1g') # Caramelo Remix by Ozuna, Karol G, and Myke Towers
```

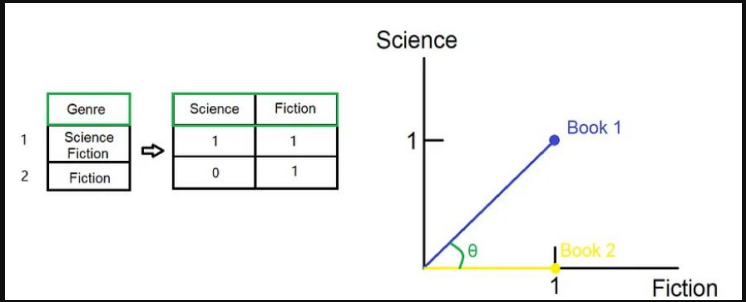
	valence	year	acousticness	artists	danceability	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	name	popularity	release_date	speechiness	t
155550	0.586	2020	0.0605	['Ozuna', 'KAROL G', 'Myke Towers']	0.755	0.772	0	67jvGGbJmOmVonlyX3mNkV	0.000092	3	0.223	-4.539	0	Caramelo - Remix	0.78	2020-09-04	0.154	16
108332	0.552	2018	0.0165	['J Balvin']	0.747	0.740	0	2D3z17LBMJ2HEHeBFFjTLi	0.000099	4	0.101	-4.325	0	Reggaeton	0.72	2018-11-17	0.191	17
19758	0.668	2020	0.0334	['J Balvin']	0.720	0.669	0	5O0zJqSMxigByaQVpMiWxX	0.000000	4	0.147	-6.244	0	Tranquila	0.01	2020-11-20	0.207	17
19745	0.668	2020	0.0334	['J Balvin']	0.720	0.669	0	6pxcVCvNr5fE58r5vxxzhF	0.000000	4	0.147	-6.244	0	Tranquila	0.06	2020-11-20	0.207	17
19768	0.668	2020	0.0334	['J Balvin']	0.720	0.669	0	70gTTmIz0WplVuPtWupcJp	0.000000	4	0.147	-6.244	0	Tranquila	0.00	2020-11-20	0.207	17

Content-based Filtering

Cosine Similarity

A measure of Vector Similarity

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



Intuition for Content-based Recommendation: Proposal One

Rationale: Lyrics are rich in information and can capture the thematic and emotional essence of a song.

BERT for Embeddings: Represent words as vectors for computing

Enhancing Embeddings: Adding metadata like genre, artist, and track name can provide additional context that isn't captured by lyrics alone.

Result: A lookup table allows for efficient retrieval of song vectors using unique tracking IDs. Ensure that this table is optimized for quick access, especially as your dataset grows.

Content-based Filtering

Slight Problem with Proposal One

- Lyrics may be hard to access
- Spotify API does not support training machine learning models starting 2023
- Numeric features not incorporated

TF-IDF

Term Frequency (TF): how frequently a term appears in a song

Inverse Document Frequency (IDF): how important a term is across the entire corpus

By assigning TF score x IDF score for dimensions vector representation

Implementation:

Use TfidfVectorizer from libraries such as scikit-learn to convert processed lyrics into a TF-IDF matrix.

If dimensionality is a problem we will just focus on track name and artist name

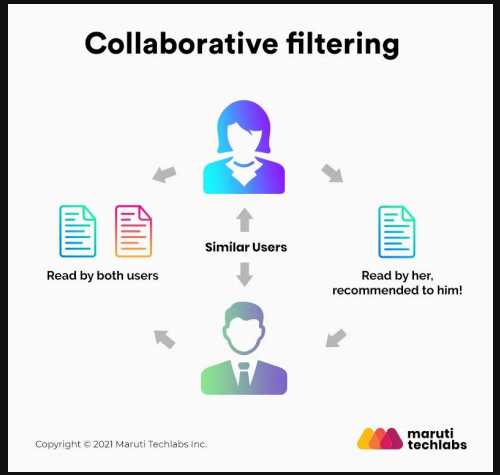
Create Numeric Feature (EX: far) use release date

Combine the TF-IDF matrix with the normalized numeric features.

Collaborative Filtering

A recommender system technique used by most recommendations systems, which groups users based on their item rating and gives recommendations based on the groups each user falls in

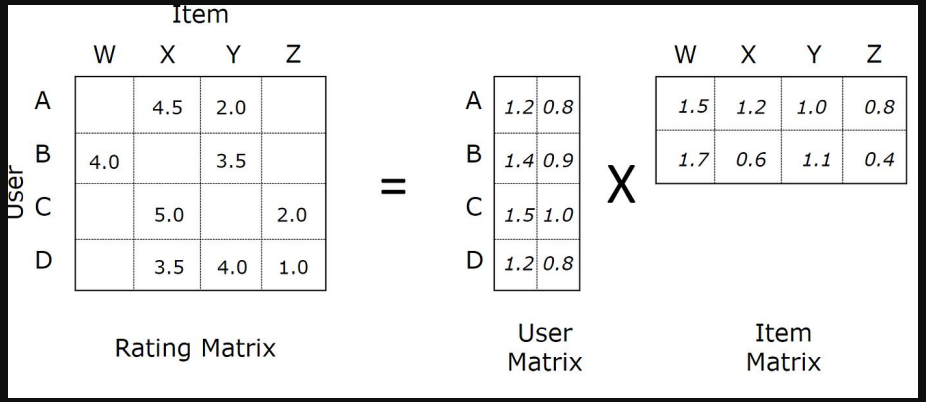
e.g., Both user A and user B enjoy items X and Y. We hypothesize that user A and B have similar preferences. If we further know that user B enjoys item Z, we can recommend item Z to user A.



Collaborative Filtering

Matrix Factorization

- Build user-item interaction
- Decompose the rating (resulting) matrix to user matrix and item matrix
- **Objective:** Identify the latent factors for user matrix and item matrix
- Determine the user weights and item weights
- Making predictions by recreating rating matrix



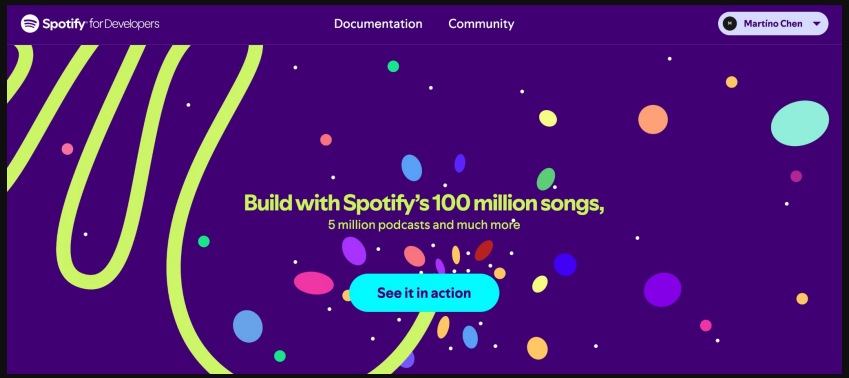
Spotipy

Spotipy is the Python library for the Spotify Web API

We definitely navigated Spotipy, managed to access to my playlists

To our knowledge, Spotify developer terms version 9 (as of May 8, 2023) disallows any content of its API to train machine learning models or AI:

- No user rating for matrix factorization
- Need additional user-item interaction data




```
client_id = "6448587d422647a8887bfba66e70067f" # Replace
client_secret = "62f6ce053995469c8[illegible]d787b84eaf4172" # Replace
redirect_uri = "http://localhost:8881" # Replace
scope = "user-library-read playlist-modify-public"

# Use SpotifyOAuth for user authorization
auth_manager = SpotifyOAuth(client_id=client_id,
                            client_secret=client_secret,
                            redirect_uri=redirect_uri,
                            scope=scope
                            )

sp = spotipy.Spotify(auth_manager=auth_manager)
```

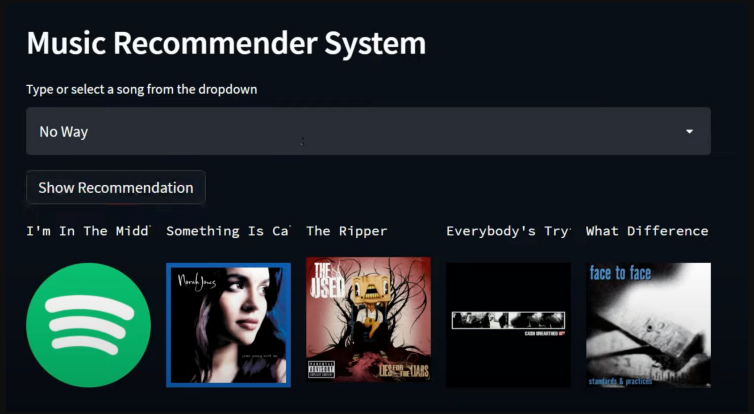
```
# Fetch user's playlists
playlists = sp.current_user_playlists(limit=10)

# Extract tracks from playlists
for playlist in playlists['items']:
    print(f"Playlist: {playlist['name']}, Total Tracks: {playlist['tracks']['total']}")
    playlist_tracks = sp.playlist_tracks(playlist['id'])
    for item in playlist_tracks['items']:
        track = item['track']
        print(f" - Track: {track['name']} by {track['artists'][0]['name']}")
```

```
Playlist: , Total Tracks: 33
 - Track: 空も飛べるはず by SPITZ
 - Track: ロビンソン by SPITZ
 - Track: 名もなき詩 by Mr.Children
 - Track: Tomorrow never knows by Mr.Children
 - Track: 異邦人 by Saki Kubota
 - Track: Akagi blues by Noboru Kirishima
 - Track: 少女A - 2012 Remaster by Akina Nakamori
 - Track: 十戒(1984) by Akina Nakamori
 - Track: 青い珊瑚礁 by Seiko Matsuda
```

Next Steps

- DBSCAN and/or Affinity PCA for Clustering
- Perform Content-based Filtering for data_by_artist.csv and data_w_genres.csv
- No user data for user-item interaction and therefore matrix factorization
  - Solution: Use Alternate data
  - e.g., Million Song Dataset Challenge
    https://www.kaggle.com/competitions/msdchallenge
- User Interface (Streamlit)
- Continue exploring Spotipy...



Thanks!