ECS 170
Group #19

Music Recommendation System

Vanessa Garcia, Yacob Kidane,
Jaishree Ramamoorthi, Nico Del Bonta

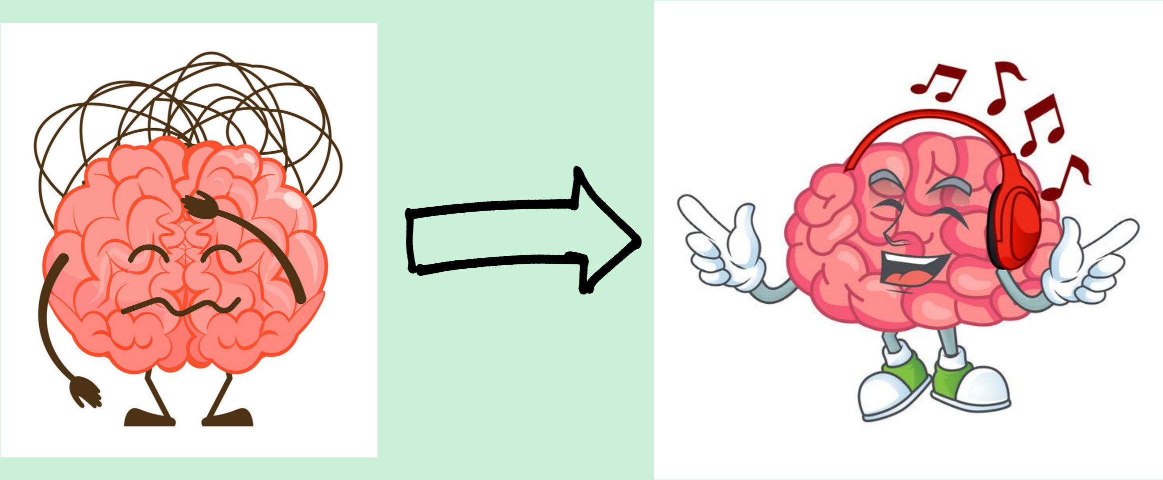
Introduction

Project Goal:

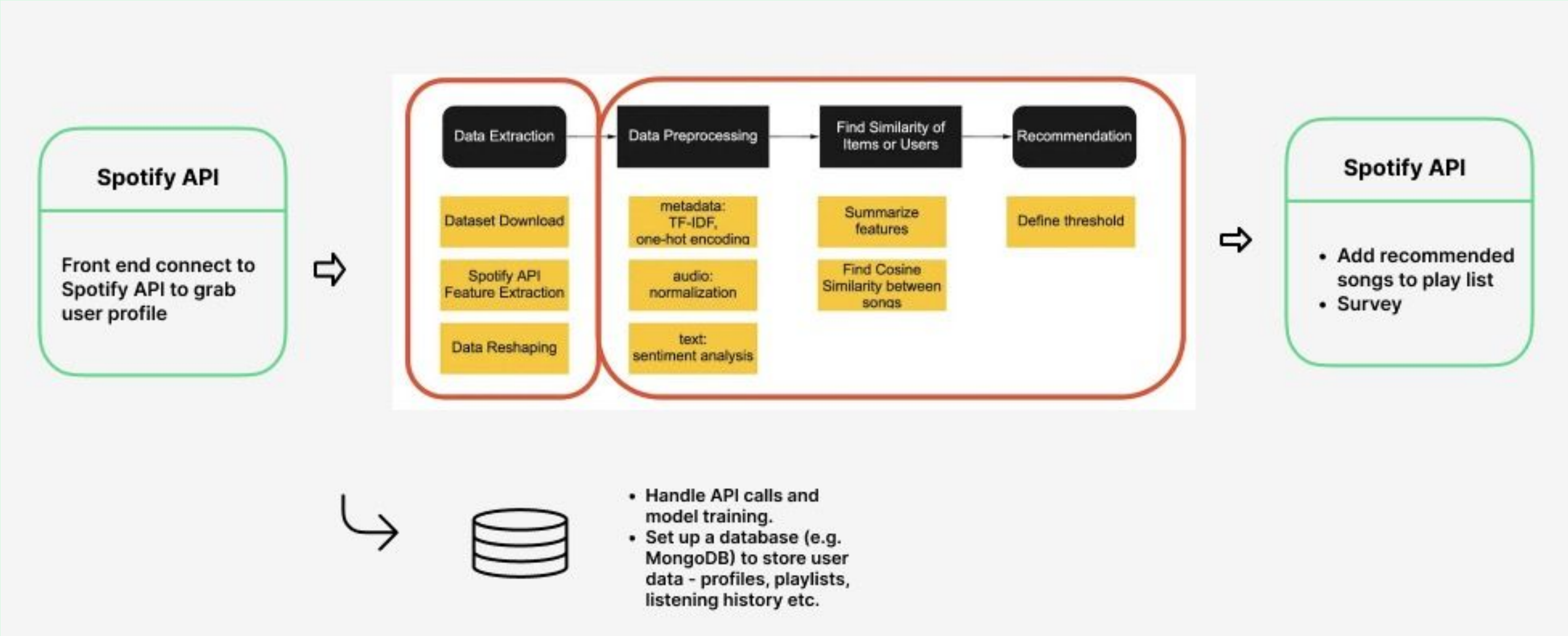
- Create a music recommendation system
  - Input: User’s top 5 favorite songs
  - Output: 5-10 songs they will likely enjoy and add to a playlist

Motivations:

- Make discovering new music easier and more time-efficient
- Reduce decision fatigue with a vast selection of music

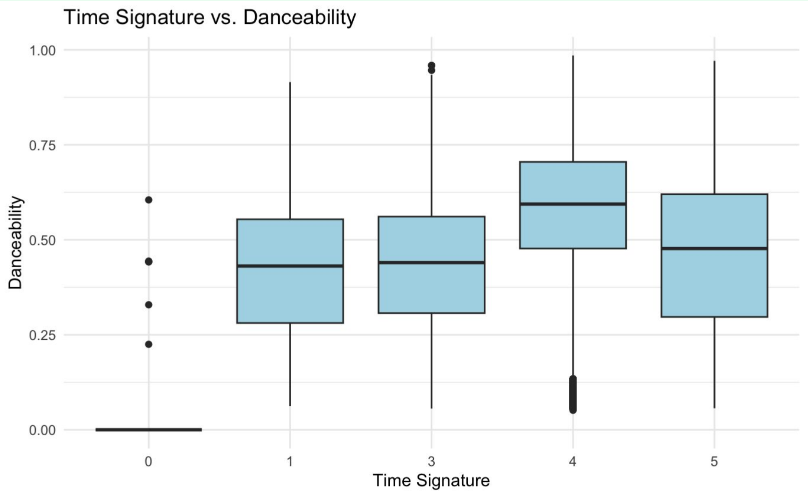


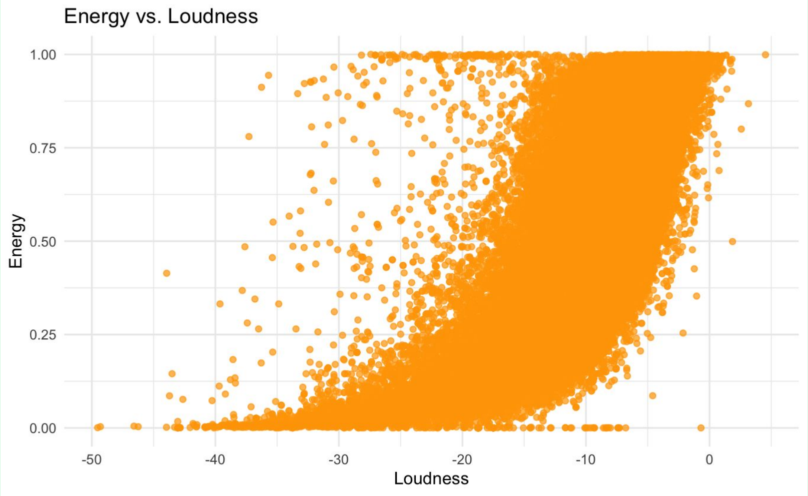
Ideal Workflow

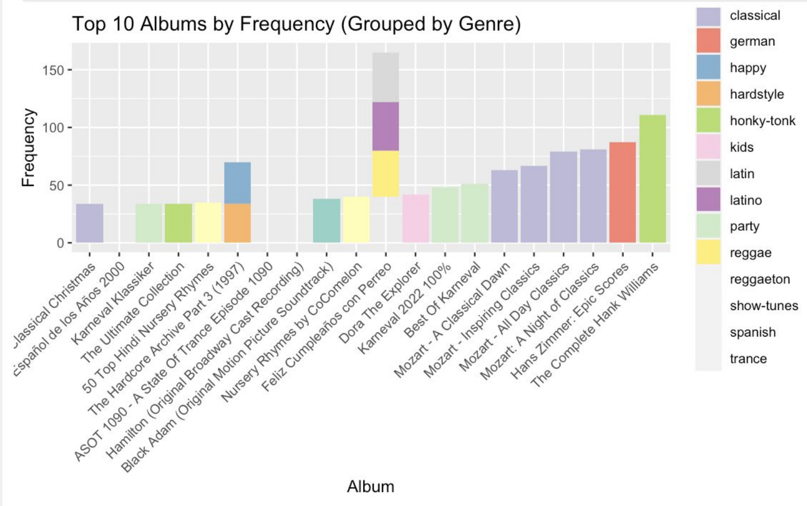


Tidying Data + Initial EDA

- [Kaggle dataset]
- Used R to clean dataset







Correlation Matrix

row_var <chr>	col_var <chr>	cor_val <dbl>
loudness	energy	0.7616900
acousticness	energy	0.7339063
loudness	acousticness	0.5898027
valence	danceability	0.4773412
instrumentalness	loudness	0.4334769
instrumentalness	valence	0.3243123
loudness	valence	0.2798479

Average Metrics Per Artist and Per Genre

Shiny App

Spotify Music Analysis

Select Comparison Type:

Compare Artists

Dataset Description

Dataset Description:
This dataset contains information on Spotify tracks from various genres. Each track is associated with audio features and is provided in CSV format. The dataset can be used for building recommendation systems, classification tasks based on audio features and genres, and other data-driven applications.
Original Kaggle Dataset: Link

App Purpose

App Purpose:
This app allows users to interact with Spotify track data and compare artists and genres based on various audio characteristics. Use the dropdown to select whether you want to compare artists or genres, and explore average metrics for each. The data is organized for easy analysis.

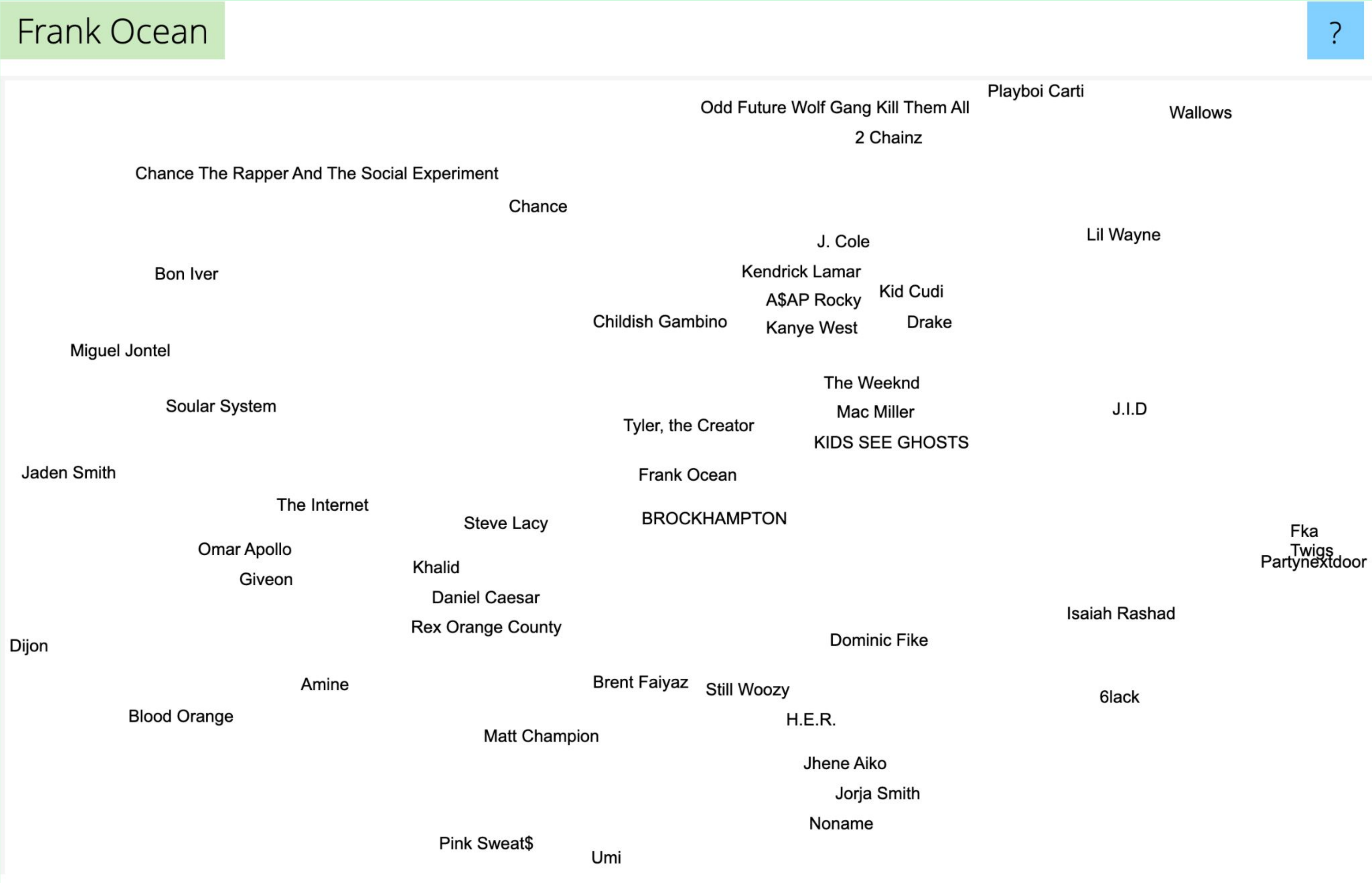
Average Metrics by Artist | Average Metrics by Genre

Show 10 entries Search:

	artists	avg_popularity	avg_duration_ms	avg_danceability	avg_energy	avg_key	avg_loudness	avg_mode	avg_speechiness	avg_acousticness	avg_instrumentalness	avg_liveness	avg_valence	avg_tempo	avg_time_signature
1	-M-	35.75	219586.375	0.586625	0.60175	7.25	-9.585	0.375	0.0821875	0.40955	0.0149304875	0.337025	0.42525	104.326125	3.875
2	:Wumpscut:	20	281493	0.533	0.778	1	-7.344	1	0.0321	0.0125	0.683	0.649	0.286	100.002	3
3	!nvite	23	137367.5	0.8205	0.519	6.5	-10.173	0.5	0.27	0.499	0.01136	0.1285	0.415	84.997	4
4	? & The Mysterians	40	176866	0.65	0.562	7	-5.965	1	0.0345	0.0623	0.0000308	0.076	0.88	123.674	4
5	¿Téo?	61	203777	0.715	0.584	0	-9.004	1	0.136	0.291	0.00366	0.068	0.429	97.214	4
6	'Bat Out Of Hell' Original Cast	24	173840	0.333	0.91	6	-1.49	0	0.0591	0.272	0	0.0891	0.647	116.055	3
7	'Be More Chill' Ensemble	41	122966.5	0.5825	0.5225	6	-7.568	1	0.241	0.6465	0.0001115	0.3065	0.694	145.0855	4
8	'Dogfight' Original Cast	25	201800	0.371	0.247	8	-10.814	1	0.037	0.873	0.00000246	0.298	0.312	121.691	4

Map of Artists

[Music Map]



Methodology:

Music Preference Survey:

purpose: gather user data

- whether the user wants explicit songs or not
- user's favorite genre
- audio metric preferences
    - using a 5 point likert scale
- list of five songs

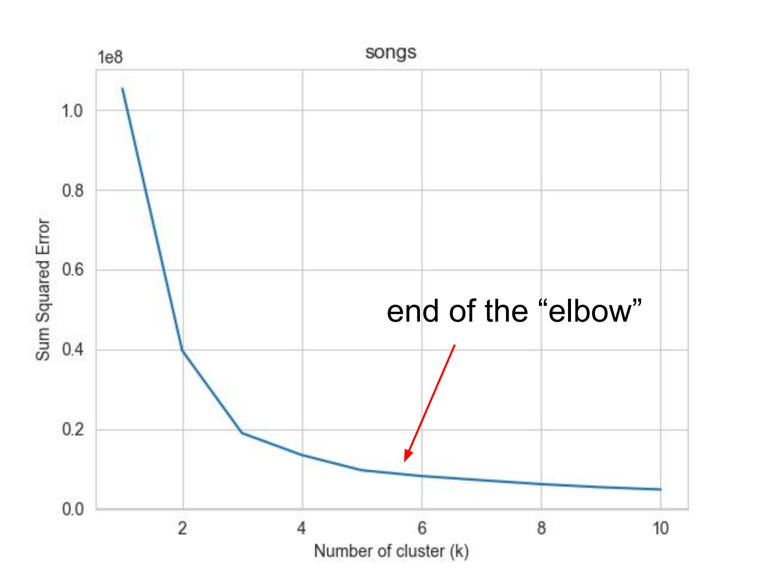
```
Would you like to filter out songs with explicit content?
* Y
* N


answer: n

Good!
------------

Please answer the following question by choosing one of the following options:

There's a lot of different genres! Would you like to see only the most popular ones?
* Y
* N


answer: y

Great choice!
------------

Please answer the following question by choosing one of the following options:

Pick a genre from the list!
* alternative
* blues
* classical
* dubstep
* country
* edm
* electronic
* hip-hop
* house
* indie-pop
* indie
* j-pop
* jazz
* k-pop
* latin
* latino
* metal
* pop
* punk-rock
* r-n-b
* reggae
* rock
* soul
* techno


answer: pop

I'll get right on that!
------------
```

only part of the survey

K-Mean Clustering:

Clustering: (Elbow Method):

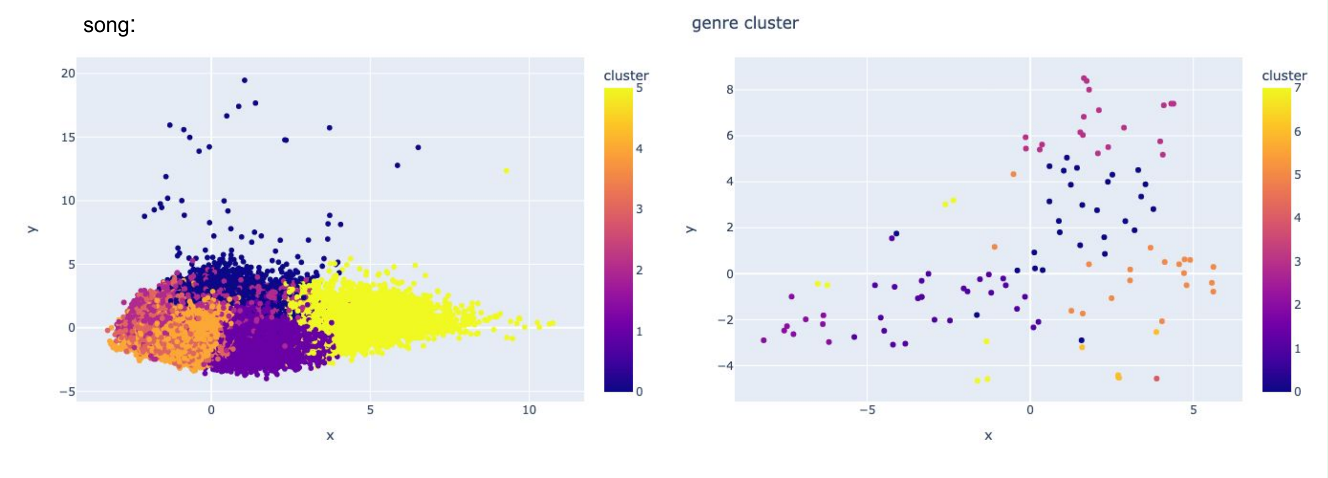
- **elbow method**
  - used to determine the optimal number of clusters for our dataset
    - genre: 8 cluster groups
    - songs: ~5-6 cluster groups



Clustering: (K-Means)

steps for each iteration:

1. put down n random center data points
2. calculate the euclidean distance between the data and each center
   a. closest center → cluster label
3. repeat



General Music Recommender (Overview):

uses three methods:

1. clusters each song in the song list and finds similar songs
2. assigns the user’s audio preferences to a song cluster
3. calculates the mean vector of the song list then cluster it

all three methods use:

- cosine similarity
- filtering

Cosine Similarity:

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

purpose: finds the similarity between two songs based on their audio features

Filtering:

1. **popularity threshold:**
    a. dataset average ~33
    b. threshold = 55
2. **genre list:**
    a. only includes songs whose genres in the accepted genre list
        i. based on clustering
3. **weighted features:**
    a. each audio features are weighted based off the correlation matrix
        i. features with higher correlations → higher weights
        ii. each rec has a score
4. **explicit songs:**
    a. if user specified to do so

Example of A Recommendation:

```
user_profile = ['N', 'POP', ['1', '1', '1', '1', '1', '1', '1', '1', '1'],
                [['Come As You Are', 'Nirvana'], ['Smells Like Teen Spirit', 'Nirvana'],
                ['Lithium', 'Nirvana'],['All Apologies', 'Nirvana'], ['Stay Away', 'Nirvana']]]
```

```
Here is your recommendations!

Based on your song list:
---------------------------------
1. Stupid for You by Waterparks
2. New Divide by Linkin Park
3. Have A Nice Day by Bon Jovi
4. Nothing For Free by Pendulum
5. Want You by Kanine
6. Glass Heart by Caskets

Based on your audio preferences:
---------------------------------
1. Guardian angel by XXXTENTACION
2. Do You? by TroyBoi

Based on the mean vector of your song list:
---------------------------------
1. Soldier Of Fortune - 2009 Digital Remaster by Deep Purple
2. Fall for You (Acoustic) by Secondhand Serenade
```

```
Here is your recommendations!

Based on your song list:
------------------------------------
1. Catastrophist by Trivium
2. Getaway - Koven Remix by Tritonal;Angel Taylor;Koven
3. Alone by I Prevail
4. Low by Wage War
5. It's Over When It's Over by Falling In Reverse
6. Hold Me Now by Caskets

Based on your audio preferences:
------------------------------------
1. Why We Thugs by Ice Cube
2. Caile by Luar La L

Based on the mean vector of your song list:
------------------------------------
1. Lost in a Wave by LANDMVRKS
2. Hold Me Now by Caskets
```

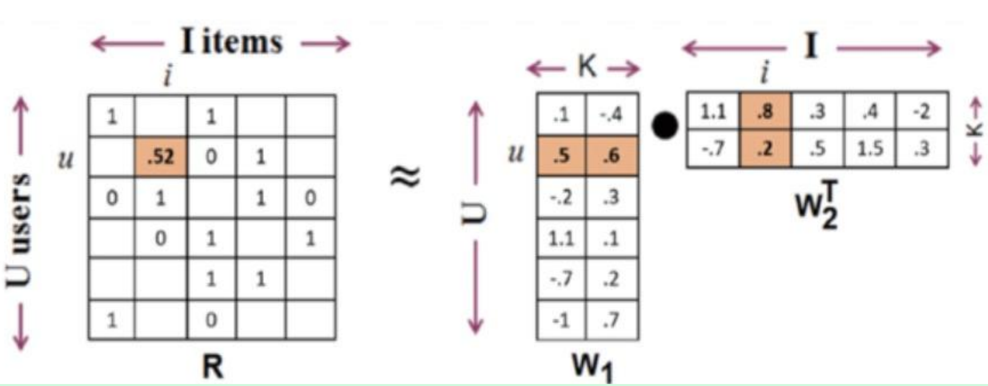
Model 2

Item Based Collaborative filtering for artists

Item Based Collaborative Filtering using a KNN

- This method utilizes preferences and behaviors of other users to come up with recommendations. The general operation of these systems is to pair users with similar tastes together into groups. Recommendations are then made based on the collective preferences of the users within each group.

- This model uses item-based recommendations are where you use the average of the k most similar items that a user has rated to suggest a rating for an item they haven't yet seen.



Implementation:

1. Imported and Preprocessed spotify Dataset
2. Pivoted clean data to KNN model
    a. Pivoting features(artist_id/songs_id and user_id with their ratings as value) of dataframes and returning a sparse matrix
3. Fitting the KNN model and creating a sparse matrix, and relating the id with the name (artist or song)
4. Recommendation logic:
    a. Uses KNN model to find the k nearest neighbours -> creates recommendations and reverse maps to fetch the values from recommendations
5. Item based Collaborative filtering
    a. Adding collaborative filtering for artist recommendation from "artist_user_ratings.csv"
6. Evaluation Metric
    a. Implemented evaluation metric for artists recommendation using “Precision Recall Metric”

Output for my profile:

```
You have input: Yacob Kidane



Gathering Recommendations for Yacob Kidane..........


Recommendations for  Yacob Kidane:
1 : Kaytranada  is similar to your favourite by  0.36771534510439996
2 : 070 Shake  is similar to your favourite by  0.3312282159485842
3 : Tame Impala  is similar to your favourite by  0.31704997464684825
4 : Sampha  is similar to your favourite by  0.2830341820895885
5 : Jordan Ward  is similar to your favourite by  0.2746874472824472
6 : Amine  is similar to your favourite by  0.2599669050343385
7 : EARTHGANG  is similar to your favourite by  0.2582208057941666
8 : Smino  is similar to your favourite by  0.228403716479746
9 : Q  is similar to your favourite by  0.20506489314898502
10 : Dijon  is similar to your favourite by  0.18025474752249182
```

```
In [11]:
favourite = 'Take Down'
df_songs = pd.read_csv('songs_users_ratings.csv')
num_recomm= 10
result_users_rec_songs = call_collaborative_KNN(df_songs, favourite,'songs',num_recomm)
```

Pros & Cons

K-Means:

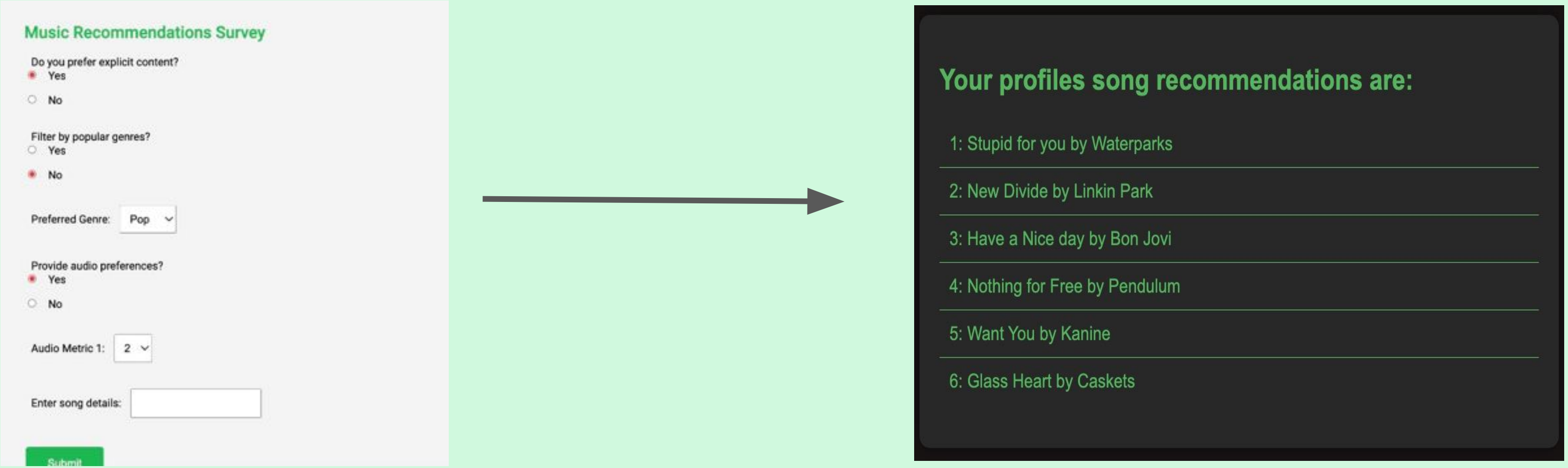
- Pros : simple/easy to use, good for unlabeled and big datasets
- Cons: inconsistent (i.e. clusters change each time)

Collaborative filtering:

- Pros:
  - Leverages user's past behaviours to recommend similar artists/songs that a user might like
- Cons:
  - May recommend "obvious" suggestions like remixes of songs and/or "covers"
  - May recommend artist that are members within a group/band

Discussion & Potential Iterations

- Limited time (~ 6 weeks)
- Improvement to model (fine tuning, reworking algorithm)
- New models and can combine models for a robust system
- Front-end development (user interface, webpage, etc.)



Conclusion

Results

- Console application
- Final product recommends songs based on song choices

Concluding Statements

- Useful in discovering new music related to user’s interests
- Unable to accurately determine efficacy of model due to variability of clusters and user preferences
- Combining models may yield more comprehensive results
    - Model 1 focuses on relating songs together
    - Model 2 focuses on relating artists together