MUSIC RECOMMENDATION SYSTEM

HAVING CONTENT-BASED AND COLLABORATIVE-FILTERING

SARTHAK JAIN (2K19/ME/215)
HARSHVARDHAN SINGH (2K19/ME/103)

INTRODUCTION

- In this project, we have built a 'Recommendation' engine, which recommends music tracks to a user, by analyzing through a machine learning algorithm.
- It is achieved through content-based and collaborative-filtering systems.
- The **Content-based system (CBRS)** recommends items based on its features and the similarity between elements of other items. Assume a user has already seen a movie from the genre of Comedy, CBRS will recommend movies that also belong to the Comedy genre.
- User preferences and attitude is considered to create **Collaborative-filtering system (CFRS)**. CFRS recommends items like what the user has already chosen.
- Collaborative Filtering is the process of filtering or evaluating items using the opinions of other people.

IMPLEMENTATION

- The dataset used to train the model is the "Million song dataset", which has around 2 million observations.
- Firstly, the dataset has been imported to a pandas dataframe. Then using functions available through the pandas library, the dataframe has been grouped by the parameter 'listen_count'.
- The dataframe obtained after the grouping can be sorted in the descending order to have insights on listening pattern of users and to build a **'popular playlist'**.
- The song_data dataframe is then used to buid a **cooccurence matrix**, using the result of which the model recommends songs to the user.

THE CO-OCCURENCE MATRIX

- A correlation matrix is a table containing correlation coefficients between variables. Each cell in the table represents the correlation between two variables.
- The value lies between -1 and 1.
- It is at the core of the Collaborative-filtering recommendation system.
- The function construct_cooccurence_matrix() is passed a list of all the songs listened by the user (user_songs).
- Then we make a list of the users, who have also listened to those songs.
- It then calculates the similarity between the songs which the user has listened to in the past and all other unique songs(all_songs).
- It does so by taking a unique song and then finding the interesection of users, who listen to the unique song and one of the (user_songs).
- For every iteration, it calculates the value of similarity to be store in the cooccurence matrix using **Jaccard Index**.

RESULTS

- Using methods provided by the pandas library, we were able to curate a 'Popular Playlist'.
- The model was able to recommend songs to an experimental user based on the songs he has listened to earlier.
- The model was able to find the similarity coefficient between two songs and then recommend songs based on that.
- The model was able to recommend unique songs to the experimental user by building a cooccurence matrix from the intersections of other user's listening preferences, (CBRS).

RESULTS

```
# recommend similar songs to the user based on his/her listening history
mcr.recommend(song_data['user_id'][40])
```

```
No. of unique songs for the user: 45
no. of unique songs in the training set: 9953
Non zero values in cooccurence_matrix :268460
```

	user_id	song	score	rank
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Quiet Houses - Fleet Foxes	0.034172	1
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Meadowlarks - Fleet Foxes	0.033473	2
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Heard Them Stirring - Fleet Foxes	0.032683	3
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Great Indoors - John Mayer	0.032123	4
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Tiger Mountain Peasant Song - Fleet Foxes	0.031740	5
5	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Sun It Rises - Fleet Foxes	0.031253	6
6	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Your Protector - Fleet Foxes	0.030409	7
7	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Oliver James - Fleet Foxes	0.030237	8
8	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Belle - Jack Johnson	0.028708	9
9	b80344d063b5ccb3212f76538f3d9e43d87dca9e	If I Could - Jack Johnson	0.028350	10

USER-BASED COLLABORATIVE FILTERING

```
# finds similar songs based on the entered song
mcr.get_similar_items(['Stacked Actors - Foo Fighters'])
```

```
no. of unique songs in the training set: 9953
Non zero values in cooccurence_matrix :5078
```

	user_id	song	score	rank
0		Generator - Foo Fighters	0.299539	1
1		Breakout - Foo Fighters	0.210863	2
2		Next Year - Foo Fighters	0.209016	3
3		Weenie Beenie - Foo Fighters	0.160550	4
4		X-Static - Foo Fighters	0.157143	5
5		For All The Cows - Foo Fighters	0.150628	6
6		Floaty - Foo Fighters	0.148760	7
7		No Way Back - Foo Fighters	0.148325	8
8		Oh_ George - Foo Fighters	0.146789	9
9		Low - Foo Fighters	0.142857	10

CONTENT-BASED FILTERING

THANK YOU!