Recommending songs

In this module, we focused on building recommender systems to find products, music and movies that interest users. In this iPython notebook we are interested in recommending songs, which compares the simple popularity-based recommendation with a personalized model, and showed the significant improvement provided by personalization.

In this assignment, we are going to explore the song data and the recommendations made by our model. In the process, you are going to learn how to use one of the most important data manipulation primitives, *groupby*.

Follow the rest of the instructions on this page to complete your program. When you are done, ***instead of uploading your code, you will answer a series of quiz questions*** (see the quiz after this reading) to document your completion of this assignment. The instructions will indicate what data to collect for answering the quiz.

Learning outcomes

- Execute song recommendation code with the iPython notebook
- Load and transform real, song data
- Build a song recommender model
- Use the model to recommend songs to individual users
- Use groupby to compute aggregate statistics of the data

Resources you will need

Download the data and starter code

Before getting started, you will need to download the dataset and the starter iPython notebook that we used in the module.

- Download the music listening dataset with articles on famous people here in csv format: **song_data.csv**
- Download the song recommendation notebook from the module here: Song recommender.ipynb

Now you are ready to get started!

What you will do

Now you are ready! We are going do three tasks in this assignment. There are several results you need to gather along the way to enter into the quiz after this reading.

1 - Counting unique users: The method .unique() can be used to select the unique elements in a column of data. In this question, you will compute the number of unique users who have listened to songs by various artists. For example, to find out the number of unique users who listened to songs by 'Kanye West', all you need to do is select the rows of the song data where the artist is 'Kanye West', and then count the number of unique entries in the 'user_id' column. Compute the number of unique users for each of these artists: 'Kanye West', 'Foo Fighters', 'Taylor Swift' and 'Lady GaGa'. Save these results to answer the quiz at the end.

2 - Using groupby-aggregate to find the most popular and least popular artist: each row of song_data contains the number of times a user listened to particular song by a particular artist. If we would like to know how many times any song by 'Kanye West' was listened to, we need to select all the rows where 'artist'=='Kanye West' and sum the 'listen_count' column. If we would like to find the most popular artist, we would need to follow this procedure for each artist, which would be very slow.

3 - Using groupby-aggregate to find the most recommended songs: Now that we learned how to use .groupby() to compute aggregates for each value in a column, let's use to find the song that is most recommended by the personalized_model model we learned in the iPython notebook above. Follow these steps to find the most recommended song:

- Split the data into 80% training, 20% testing, using seed=0, as was done in the iPython notebook above.

- Train an item_similarity_recommender, as done in the iPython notebook, using the training data.

Next, we are going to make recommendations for the users in the test data, but there are over 200,000 users (58,628 unique users) in the test set. Computing recommendations for these many users can be slow in some computers. Thus, we will use only the first 10,000 users only in this question.

By sorting the results, you will find out the most recommended song to the first 10,000 users in the test data! Due to randomness in train-test split, the most recommended song may come out differently for different people.