Song Recommendation System

Content Based Recommendation System

# Domain

Music, Digital Media

# Context

A lot of companies worldwide recommend songs to listeners based on their interests. Some popular examples of such companies are Spotify, iTunes, Gaana and Saavn. Song recommendations helps users to discover new artists that make music similar to the genre which they listen to. This helps in increasing revenue across these platforms and helps artists make a survival by streaming their music online.

Approach

As part of this exercise we will build a recommendation system that recommends a list of songs based on the user's song preference.

There are 2 files that we will be using in this case study, 'songs.csv' and 'song\_extra\_info.csv'.

Attribute Information

* The 'songs.csv' file has the following attributes:
* song\_id: Unique id of the song
* song\_length: Duration of the song
* genre\_ids: Unique id of the genre of the song
* artist\_name: Name of the artist who represents the song
* composer: Name of the composer of the song
* lyricist: Name of the lyricist of the song
* language: The language of the song

The 'song\_extra\_info.csv' file has the following attributes:

* song\_id: Unique id of the song
* name: name of the song
* isrc: International standard recording code

Steps

1. Read the 2 files songs.csv and song\_extra\_info.csv and explore the shape of the dataframes and the content and columns in it
2. Merge the 2 files into 1, drop all columns except ‘song\_id’, ‘name’ and ‘composer’ and check for any missing values in these columns and drop them
3. Sample 10000 data points from the entire dataset and use random\_state=98 for reproducibility of data samples
4. Use tfidfvectorizer from sklearn and fit it onto the composer column and generated the term document frequency matrix
5. Calculate the cosine similarity for this matrix. Create an indices dataframe from the original data frame that only holds the song and it’s id information. (i.e the new ids as per the newly generated sample dataframe)
6. Write a function that takes in a song name as an argument, finds it's index. Then it gets a list of all similarity scores for the song index. Then it sorts the similarity scores from highest to lowest and takes only the first 30 scores and returns the song names for these indices with highest scores.
7. Test out your function by sending it a song name from the data sampled

Learning Outcomes

* Exploratory Data Analysis - dealing with missing data
* Content based recommendation systems
* Term frequency inverse document frequency(tfidf)