MUSIC RECOMMENDATION SYSTEM

**Dataset Information**

The dataset contains two files

* triplet\_file
* metadata\_file.

**The triplet file** contains **user\_id, song\_id**, and **how many times listen to the song.**

**The metadata file** contains **song\_id, title, release, artist\_name** and **year**

The Dataset is a mixture of the song from the various website with the rating that users gave after listening to the song.

There are 3 types of recommendation systems

**Content-based**

Content-based filtering is a type of recommender system that attempts to guess what a user may like based on that user's activity. Content-based filtering makes recommendations by using keywords and attributes assigned to objects in a database (e.g., items in an online marketplace) and matching them to a user profile.

**Collaborative**

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users.

It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user. It looks at the items they like and combines them to create a ranked list of suggestions.

There are many ways to decide which users are similar and combine their choices to create a list of recommendations. This article will show you how to do that with Python.

**Popularity**

It is a type of recommendation system which works on the principle of popularity and or anything which is in trend. These systems check the product or movies which are in trend or are most popular among the users and directly recommend those.

**Import Libraries**

import pandas as pd

import numpy as np

**Change the directory**

import the custom module created for recommendation **- Recommenders**

import os

os.chdir ('C:\\Noble\\Training\\Acmegrade\\Machine Learning\\Projects\\PRJ Music Recommendation System\\')

import Recommenders as Recommenders

**Load the Data -** triplets\_file

song\_df\_1 = pd.read\_csv('triplets\_file.csv')

print (song\_df\_1.shape)

print (song\_df\_1.head())

**Load the Data -** song\_data

song\_df\_2 = pd.read\_csv('song\_data.csv')

print (song\_df\_2.shape)

display (song\_df\_2.head())

**Combine two data frames and create one data Frame**

Common column: Song Id

Remove any duplicate song id in the data frame - song\_df\_2.

This is to reduce inflated results after joining data frames

song\_df = pd.merge(song\_df\_1, song\_df\_2.drop\_duplicates(['song\_id']), on='song\_id', how='left')

display (song\_df.shape)

display (song\_df.head())

**Display the length of each data Frame**

print(len(song\_df\_1), len(song\_df\_2))

**Length of the consolidated data frame**

len(song\_df)

**Select only 50000 records to create a model**

This is to improve performance

song\_df = song\_df.head(50000)

song\_df.shape

**Creating new feature combining title and artist name**

song\_df['song'] = song\_df['title']+' - '+song\_df['artist\_name']

song\_df.head()

**Cumulative sum of listen count of the songs**

**Group by based on Song**

song\_grouped = song\_df.groupby(['song']).agg({'listen\_count':'count'}).reset\_index()

song\_grouped.head()

**Sum of count – This is same as total number of records**

grouped\_sum = song\_grouped['listen\_count'].sum()

grouped\_sum

**Display the percentage to identify most popular song**

song\_grouped['percentage'] = (song\_grouped['listen\_count'] / grouped\_sum ) \* 100

song\_grouped.sort\_values(['listen\_count', 'song'], ascending=[0,1])

Popularity Recommendation Engine

**Import Popularity Recommender Model**

pr = Recommenders.popularity\_recommender\_py()

pr.create(song\_df, 'user\_id', 'song')

**Display the top 10 popular songs- User 5**

pr.recommend(song\_df['user\_id'][5])

**Display the top 10 popular songs- User 100**

This is popularity rating same for all users

pr.recommend(song\_df['user\_id'][100])

Item Similarity Recommendation

**Import Item Similarity Model**

ir = Recommenders.item\_similarity\_recommender\_py()

ir.create(song\_df, 'user\_id', 'song')

**Item Similarity for User -5**

user\_items = ir.get\_user\_items(song\_df['user\_id'][5])

# display user songs history

for user\_item in user\_items:

print(user\_item)

**Item Similarity for User -100**

user\_items = ir.get\_user\_items(song\_df['user\_id'][100])

# display user songs history

for user\_item in user\_items:

print(user\_item)

**Give song recommendation for that user-5**

ir.recommend(song\_df['user\_id'][5])

**Give song recommendation for that user-100**

ir.recommend(song\_df['user\_id'][100])

Recommendation based on Song Name

**Based on selected song provide recommendation**

ir.get\_similar\_items(['Oliver James - Fleet Foxes', 'The End - Pearl Jam'])

**Recommendation for another song**

ir.get\_similar\_items(['Use Somebody - Kings Of Leon'])