Addis Ababa University
አዲስ፡አበባ፡ዩኒቨርሲቲ
Seek Wisdom, Elevate your Intellect and Serve Humanity

School of Information Science (Information Science Track)

Module title: Data Analytics and Visualizations

Course Code: InSS 624

Project Report

on

Data Analytic and Visualization for Online music recommendation

Prepared by:

Meresa Hiluf.....................................GSR/8123/11

Submitted to: Dr.Melkamu B.

July 6, 2019

Table of Contents

1. Introduction

The Online Music Recommender system was trying to identify songs, which are most relevant to the user. (e.g top n numbers of music offers)
In this project I use popularity based and collaborative based recommendation system to recommend the popular music by counting number of times listened and ranking them and by personalize each users interest based on their emotions respectively.
Activities covered in this project is summarized as follow.

- Download the dataset from http://github.com (two files one text file for a user's and another for songs)
- Load the dataset and merge them to one input file for analysis the recommender system
- Check the length of each file and select the subset of songs to reduce the data size and remove duplications.
- Build and test using **popularity model**, which is it can count and rank the most popular songs by counting numbers of listened.
    - Sort songs by popularly in decreasing order
    - For each user, recommend the songs in order of popularly, except those in the user's profile.
    - Its importance is, it is simple, easy to implement and serve as baseline
    - Its negative side is, it cannot personalize (users and songs information are not taken in account, only count number of listen and rank them)
    - Some song will never be listened also.
- Build collaborative based model
    - Songs that are often listened by the same user tend to be similar and are more likely to be listened together in future by some one other users.
    - Users who listen to the same songs in the past tends to have similar interests and will probably listen to the same songs in future.
    - In this model it is better recommender, it recommends different song for different users based on their interests, users with same interest will also have the probability to recommend same sons, rather than the popularity based.
- Finally, I use precision and recall evaluation method to compare both models, and based on the evaluation the collaborative based recommendation system is good one.

→ *follow the following step to check each codes and results*

2. Step by step activities

2.1.Load music dataset

Music dataset is too big, we need how to manage and give you wise suggestions according to the taste! For this project I download two datasets in text and csv file format for song and user datasets, I was working with as follow.

#the triple file contains (user_id, song_id and listen time in text format)

#songs metadata file contains song_id, song title, release_by and artist_name

Code: #to read the dataset and merge the two dataset t

```
triplets_file = 'E:/AAU IS 2019/Sem-2/Data Analytics and Visualization/1-Music recomender/triplet_file.txt'
songs_metadata_file = 'E:/AAU IS 2019/Sem-2/Data Analytics and Visualization/1-Music recomender/song_data.csv'

song_df_1 = pandas.read_table(triplets_file,header=None)
song_df_1.columns = ['user_id', 'song_id', 'listen_count']

#Read song  metadata
song_df_2 =  pandas.read_csv(songs_metadata_file)

#Merge the two data frames above to create input data frame for recommender systems
song_df = pandas.merge(song_df_1, song_df_2.drop_duplicates(['song_id']), on="song_id", how="left")
```

2.2. Explore the data

Music data shows how many times a user listened to a song, as well as the details of the song.

song_df.head(10) ***#display top how many times a user listened to a song as follow***

In [3]: `song_df.head(10)`

Out[3]:

	user_id	song_id	listen_count	song	release	artist_name	year
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1.0	The Cove	Thicker Than Water	Jack Johnson	0.0
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2.0	Entre Dos Aguas	Flamenco Para Niños	Paco De Lucia	1976.0
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1.0	Stronger	Graduation	Kanye West	2007.0
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1.0	Constellations	In Between Dreams	Jack Johnson	2005.0
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODACBL12A8C13C273	1.0	Learn To Fly	There Is Nothing Left To Lose	Foo Fighters	1999.0
5	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODDNQT12A6D4F5F7E	5.0	Apuesta Por El Rock 'N' Roll	Antología Audiovisual	Héroes del Silencio	2007.0
6	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODXRTY12AB0180F3B	1.0	Paper Gangsta	The Fame Monster	Lady GaGa	2008.0
7	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOFGUAY12AB017B0A8	1.0	Stacked Actors	There Is Nothing Left To Lose	Foo Fighters	1999.0
8	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOFRQTD12A81C233C0	1.0	Sehr kosmisch	Musik von Harmonia	Harmonia	0.0

2.3. Length of the dataset

To show total tuples in the dataset

```
In [5]: len(song_df)
Out[5]: 28981
```

2.4. Create a subset of the dataset

I create subset dataset to reduce duplicate data and reduce the data size to 12000 from the total 28981 as follow

```
song_df = song_df.head(12000)
#Merge song title and artist_name columns to make a merged column
song_df['song'] = song_df['song'].map(str) + " - " + song_df['artist_name']
```

2.5. Show the most popular songs in the dataset

Code:

```
song_grouped = song_df.groupby(['song']).agg({'listen_count': 'count'}).reset_index()
grouped_sum = song_grouped['listen_count'].sum()
song_grouped['percentage']  = song_grouped['listen_count'].div(grouped_sum)*100
song_grouped.sort_values(['listen_count', 'song'], ascending = [0,1]).head(10)
```

Output:

	song	listen_count	percentage
4107	Sehr kosmisch - Harmonia - Harmonia	51	0.425000
1191	Dog Days Are Over (Radio Edit) - Florence + Th...	38	0.316667
5253	Undo - Björk - Björk	38	0.316667
3896	Revelry - Kings Of Leon - Kings Of Leon	35	0.291667
4101	Secrets - OneRepublic - OneRepublic	34	0.283333
5733	You're The One - Dwight Yoakam - Dwight Yoakam	34	0.283333
4920	The Scientist - Coldplay - Coldplay	30	0.250000
5291	Use Somebody - Kings Of Leon - Kings Of Leon	29	0.241667
1552	Fireflies - Charttraxx Karaoke - Charttraxx Ka...	28	0.233333
2076	Horn Concerto No. 4 in E flat K495: II. Romanc...	27	0.225000

2.6. Count number of unique users in the dataset

Code:

```
users = song_df['user_id'].unique()
len(users)
```

Output:

```
442 # based on the popularity rank there are 442 unique users
```

2.7. Count the number of unique songs in the dataset

Code:

```
songs = song_df['song'].unique()
len(songs)
```

Result:

```
5780  # form all the songs there are 5780 unique songs.
```

2.8. Showing the most popular songs in the dataset

Code:

```
song_grouped = song_df.groupby(['song']).agg({'listen_count':
'count'}).reset_index()
grouped_sum = song_grouped['listen_count'].sum()
song_grouped['percentage']  =
song_grouped['listen_count'].div(grouped_sum)*100
song_grouped.sort_values(['listen_count', 'song'], ascending =
[0,1]).head(10)
```

Output:

Out[9]:

	song	listen_count	percentage
4107	Sehr kosmisch - Harmonia - Harmonia	51	0.425000
1191	Dog Days Are Over (Radio Edit) - Florence + Th...	38	0.316667
5253	Undo - Björk - Björk	38	0.316667
3896	Revelry - Kings Of Leon - Kings Of Leon	35	0.291667
4101	Secrets - OneRepublic - OneRepublic	34	0.283333
5733	You're The One - Dwight Yoakam - Dwight Yoakam	34	0.283333
4920	The Scientist - Coldplay - Coldplay	30	0.250000
5291	Use Somebody - Kings Of Leon - Kings Of Leon	29	0.241667
1552	Fireflies - Charttraxx Karaoke - Charttraxx Ka...	28	0.233333
2076	Horn Concerto No. 4 in E flat K495: II. Romanc...	27	0.225000

3. Building a Recommender Model

I use 20% of the dataset for test data set and the rest 80% for train data set. It also display the top 5 recommend data for training data.

Code:

```
train_data, test_data = train_test_split(song_df, test_size = 0.20, random_state=0)
print(train_data.head(5))
```

Output:

```
                              user_id             song_id  \
843   baf47ed8da24d607e50d8684cde78b923538640f  SORWPCP12A8C13B9D8
9450  97e48f0f188e04dcdb0f8e20a29dacb881d80c9e  SOYJETS12A8C13ECC7
7766  390c2e81bc9cf885608a0891c0a7eb13f1fd3336  SOUVTSM12AC468F6A7
9802  4b65fe3f5e0caff1cd870637d0f05be160a721c4  SOJUOYC12AB017F6A7
8555  6f8453b0d9d2199f98c1992995a8445ad6837fd8  SOLPTVW12A8C13F136

      listen_count                                         song  \
843            2.0              Tape Song - The Kills - The Kills
9450           1.0  The Body Says No - The New Pornographers - The...
7766           1.0  Drop The World - Lil Wayne / Eminem - Lil Wayn...
9802           1.0              Jailbreak - Thin Lizzy - Thin Lizzy
8555           2.0  All You Need Is Love - Jim Sturgess / Dana Fuc...

                             release             artist_name    year
843                     Midnight Boom               The Kills  2008.0
9450                    Mass Romantic    The New Pornographers  2000.0
7766                   Drop The World       Lil Wayne / Eminem     0.0
9802  The Boys Are Back In Town / Jailbreak          Thin Lizzy  1976.0
8555              Across The Universe  Jim Sturgess / Dana Fuchs     0.0
```

3.1. Popularly Based

3.1.1. Simple popularity-based recommender class (Can be used as a black box)

I use class for the simple popularity recommender class (Recommenders.popularity_recommender_py) and Create an instance of popularity-based recommender class as follow…

Code:

```
pm = Recommenders.popularity_recommender_py()
pm.create(train_data, 'user_id', 'song')

# In this case the sample predicted music for user 5 are as follow
user_id = users[5]
```

pm.recommend(user_id)

Output:

```
Out[15]:
```

	user_id	song	score	Rank
3628	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Sehr kosmisch - Harmonia - Harmonia	45	1.0
1040	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Dog Days Are Over (Radio Edit) - Florence + Th...	31	2.0
3448	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Revelry - Kings Of Leon - Kings Of Leon	29	3.0
4634	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Undo - Björk - Björk	29	4.0
5045	4bd88bfb25263a75bbdd467e74018f4ae570e5df	You're The One - Dwight Yoakam - Dwight Yoakam	28	5.0
3622	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Secrets - OneRepublic - OneRepublic	27	6.0
1364	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Fireflies - Charttraxx Karaoke - Charttraxx Ka...	23	7.0
4669	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Use Somebody - Kings Of Leon - Kings Of Leon	23	8.0
774	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Clocks - Coldplay - Coldplay	22	9.0
1778	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Hey_ Soul Sister - Train - Train	22	10.0

Even-if the popularity-based recommendation simple and easy to implement, the result for all users is the same. Because it will only count the numbers of songs listened and rank them by their descending orders e.g for the usre-ID-5 in the above and for user-ID- 6 below is the same.

```
In [41]: ###Fill in the code here, but the results is steel the same, why??
         user_id = users[6]
         pm.recommend(user_id)

Out[41]:
```

	user_id	song	score	Rank
3628	e006b1a48f466bf59feefed32bec6494495a4436	Sehr kosmisch - Harmonia	45	1.0
1040	e006b1a48f466bf59feefed32bec6494495a4436	Dog Days Are Over (Radio Edit) - Florence + Th...	31	2.0
3448	e006b1a48f466bf59feefed32bec6494495a4436	Revelry - Kings Of Leon	29	3.0
4634	e006b1a48f466bf59feefed32bec6494495a4436	Undo - Björk	29	4.0
5045	e006b1a48f466bf59feefed32bec6494495a4436	You're The One - Dwight Yoakam	28	5.0
3622	e006b1a48f466bf59feefed32bec6494495a4436	Secrets - OneRepublic	27	6.0
1364	e006b1a48f466bf59feefed32bec6494495a4436	Fireflies - Charttraxx Karaoke	23	7.0
4669	e006b1a48f466bf59feefed32bec6494495a4436	Use Somebody - Kings Of Leon	23	8.0
774	e006b1a48f466bf59feefed32bec6494495a4436	Clocks - Coldplay	22	9.0
1778	e006b1a48f466bf59feefed32bec6494495a4436	Hey_ Soul Sister - Train	22	10.0

In this case we, need other models, to personalized the recommendations, user based similarity, which mean, songs that are often listened by the same user tend to be similar and are more likely to be listened together in future by some one other users and it recommends different songs for different users based on their interest. See in section 3.2 bellow.

3.2. Personalized Recommender

Build a song recommender with personalization, just we now create an item similarity based collaborative filtering model that allows us to make personalized recommendations to each user. In this model I create a class for item-based reminder in python .py file(item_similarity_recommender_py) and the model uses K-nearest neighbor algorithm to recommend song based on their cooccurrence matrix.

Code:

```
is_model = Recommenders.item_similarity_recommender_py()
is_model.create(train_data, 'user_id', 'song')

#Print the songs for the user in training data, in this e.g for user-6
user_id = users[6]
user_items = is_model.get_user_items(user_id)

print("------------------------------------------------------------------------------------")
print("Training data songs for the user userid: %s:" % user_id)
print("------------------------------------------------------------------------------------")

for user_item in user_items:
    print(user_item)

print("----------------------------------------------------------------------")
print("Recommendation process going on:")
print("----------------------------------------------------------------------")

#Recommend songs for the user using personalized model
is_model.recommend(user_id)
```

Output for the training data

```
------------------------------------------------------------------------------------
Training data songs for the user userid: e006b1a48f466bf59feefed32bec6494495a4436:
------------------------------------------------------------------------------------
Secrets - OneRepublic - OneRepublic
Where Did You Sleep Last Night - Nirvana - Nirvana
Undo - Björk - Björk
Rhyme & Reason - DAVE MATTHEWS BAND - DAVE MATTHEWS BAND
Ain't Misbehavin - Sam Cooke - Sam Cooke
Fireflies - Charttraxx Karaoke - Charttraxx Karaoke
Horn Concerto No. 4 in E flat K495: II. Romance (Andante cantabile) - Barry
Tuckwell/Academy of St Martin-in-the-Fields/Sir Neville Marriner - Barry
Tuckwell/Academy of St Martin-in-the-Fields/Sir Neville Marriner
Esta Es Para Hacerte Féliz - Jorge Gonzalez - Jorge Gonzalez
Hey_ Soul Sister - Train - Train
Drop The World - Lil Wayne / Eminem - Lil Wayne / Eminem
Sehr kosmisch - Harmonia - Harmonia
Revelry - Kings Of Leon - Kings Of Leon
OMG - Usher featuring will.i.am - Usher featuring will.i.am
Blow Me Away - Breaking Benjamin - Breaking Benjamin
Use Somebody - Kings Of Leon - Kings Of Leon
Marry Me - Train - Train
For You (Amended/Radio Edit LP) - Staind - Staind
You're The One - Dwight Yoakam - Dwight Yoakam
Dog Days Are Over (Radio Edit) - Florence + The Machine - Florence + The Machine
Cry For Help (Album Version) - Shinedown - Shinedown
Lady In Black - Ensiferum - Ensiferum
Come As You Are - Nirvana - Nirvana
Corn Bread - DAVE MATTHEWS BAND - DAVE MATTHEWS BAND
----------------------------------------------------------------------
Recommendation process going on:
----------------------------------------------------------------------
```

```
No. of unique songs for the user: 23
no. of unique songs in the training set: 5089
Non zero values in cooccurence_matrix :8452
```

<u>Output for the Recommendation</u>

Out[25]:

	user_id	song	score	rank
0	e006b1a48f466bf59feefed32bec6494495a4436	Lucky (Album Version) - Jason Mraz & Colbie Ca...	0.081741	1
1	e006b1a48f466bf59feefed32bec6494495a4436	Bulletproof - La Roux - La Roux	0.077576	2
2	e006b1a48f466bf59feefed32bec6494495a4436	Somebody To Love - Justin Bieber - Justin Bieber	0.076084	3
3	e006b1a48f466bf59feefed32bec6494495a4436	Heartbreak Warfare - John Mayer - John Mayer	0.071851	4
4	e006b1a48f466bf59feefed32bec6494495a4436	Love Story - Taylor Swift - Taylor Swift	0.069784	5
5	e006b1a48f466bf59feefed32bec6494495a4436	Alejandro - Lady GaGa - Lady GaGa	0.066811	6
6	e006b1a48f466bf59feefed32bec6494495a4436	Just Dance - Lady GaGa / Colby O'Donis - Lady ...	0.062858	7
7	e006b1a48f466bf59feefed32bec6494495a4436	The Scientist - Coldplay - Coldplay	0.058436	8
8	e006b1a48f466bf59feefed32bec6494495a4436	Creep (Explicit) - Radiohead - Radiohead	0.054596	9
9	e006b1a48f466bf59feefed32bec6494495a4436	Bleed It Out [Live At Milton Keynes] - Linkin ...	0.053916	10

Use the personalized model to make recommendations for the following user id(5). (Note the difference in recommendations from the first user id (6).)

In [40]:
```
#Fill in the code here
user_id = users[5]
is_model.recommend(user_id)
```

```
No. of unique songs for the user: 13
no. of unique songs in the training set: 5089
Non zero values in cooccurence_matrix :2313
```

Out[40]:

	user_id	song	score	rank
0	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Mockingbird - Eminem	0.080963	1
1	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Superman - Eminem / Dina Rae	0.075293	2
2	4bd88bfb25263a75bbdd467e74018f4ae570e5df	U Smile - Justin Bieber	0.053859	3
3	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Favorite Girl - Justin Bieber	0.048596	4
4	4bd88bfb25263a75bbdd467e74018f4ae570e5df	I'm Back - Eminem	0.047798	5
5	4bd88bfb25263a75bbdd467e74018f4ae570e5df	I'm On A Boat - The Lonely Island / T-Pain	0.047192	6
6	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Here Without You - 3 Doors Down	0.046497	7
7	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Overboard - Justin Bieber / Jessica Jarrell	0.046154	8
8	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Teach Me How To Dougie - California Swag District	0.044448	9
9	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Killing In The Name - Rage Against The Machine	0.044379	10

3.3. Similar song to any song in the dataset

We can also apply the model to find similar songs to any song in the dataset. The top 10 songs similar to the song '*U Smile - Justin Bieber*' as follow.

```
In [25]: is_model.get_similar_items(['U Smile - Justin Bieber'])

no. of unique songs in the training set: 5089
Non zero values in cooccurence_matrix :197

Out[25]:
```

	user_id	song	score	rank
0		Hace Tiempo - Fonseca	0.400000	1
1		Down To Earth - Justin Bieber	0.400000	2
2		What You Know - Two Door Cinema Club	0.333333	3
3		Monster - Lady GaGa	0.300000	4
4		Paper Planes - M.I.A.	0.300000	5
5		Killing In The Name - Rage Against The Machine	0.285714	6
6		Stuck In The Moment - Justin Bieber	0.285714	7
7		Favorite Girl - Justin Bieber	0.285714	8
8		Somebody To Love - Justin Bieber	0.277778	9
9		One - Metallica	0.250000	10

4. Evaluation of the Models

4.1. Quantitative comparison between the models

We now formally compare the popularity and the personalized models using ***precision-recall*** curves. Class to calculate precision and recall can be used as a *black box* (Evaluation class)

#Evaluation.precision_recall_calculator

Code:

```
start = time.time()

#Define what percentage of users to use for precision recall calculation
user_sample = 0.05

#Instantiate the precision_recall_calculator class
pr = Evaluation.precision_recall_calculator(test_data, train_data, pm, is_model)

#Call method to calculate precision and recall values
(pm_avg_precision_list, pm_avg_recall_list, ism_avg_precision_list, ism_avg_recall_list) =
pr.calculate_measures(user_sample)

end = time.time()
print(end - start)
```

Output:

```
Length of user_test_and_training:319
Length of user sample:15
Getting recommendations for user:c24ec42f0e449ff39a95a01f0795f833b898f71b
No. of unique songs for the user: 136
no. of unique songs in the training set: 4483
Non zero values in cooccurence_matrix :38164
Getting recommendations for user:e06fdac28cdd1d69d89de27868d5f02f71c2ee44
No. of unique songs for the user: 27
no. of unique songs in the training set: 4483
Non zero values in cooccurence_matrix :5142
Getting recommendations for user:c1fc436b58e28b3e3f1b43a4e955baa19d8a69ba
```

```
No. of unique songs for the user: 13
no. of unique songs in the training set: 4483
Non zero values in cooccurence_matrix :1440

………. Continue for all user
```

4.2. Code to plot Precision Recall curve

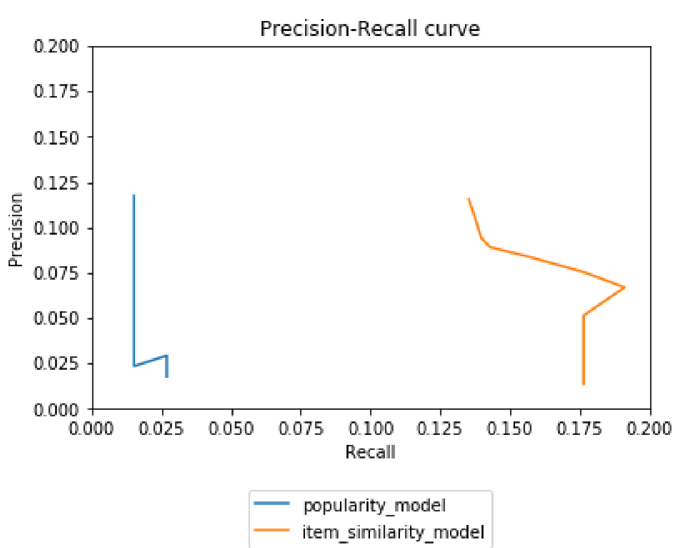
Code:

```
import pylab as pl
#Method to generate precision and recall curve
def plot_precision_recall(m1_precision_list, m1_recall_list, m1_label, m2_precision_list, m2_recall_list, m2_label):
   pl.clf()
   pl.plot(m1_recall_list, m1_precision_list, label=m1_label)
   pl.plot(m2_recall_list, m2_precision_list, label=m2_label)
   pl.xlabel('Recall')
   pl.ylabel('Precision')
   pl.ylim([0.0, 0.20])
   pl.xlim([0.0, 0.20])
   pl.title('Precision-Recall curve')
   #pl.legend(loc="upper right")
   pl.legend(loc=9, bbox_to_anchor=(0.5, -0.2))
   pl.show()

print("Plotting precision recall curves.")
plot_precision_recall(pm_avg_precision_list, pm_avg_recall_list, "popularity_model",
               ism_avg_precision_list, ism_avg_recall_list, "item_similarity_model")
```

Output:

```
Plotting precision recall curves.
```



Generate Precision Recall curve using pickled results on a larger data subset

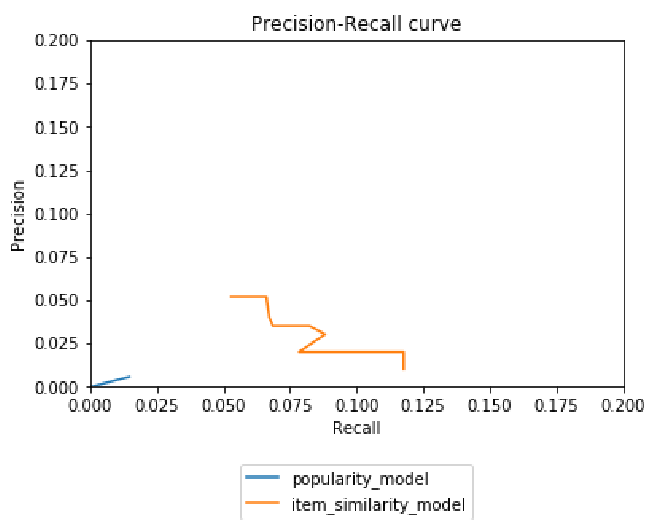
Code:

```
print("Plotting precision recall curves for a larger subset of data (12,000 rows) (user sample = 0.05).")

#Read the persisted files
pm_avg_precision_list = joblib.load('pm_avg_precision_list_3.pkl')
pm_avg_recall_list = joblib.load('pm_avg_recall_list_3.pkl')
ism_avg_precision_list = joblib.load('ism_avg_precision_list_3.pkl')
ism_avg_recall_list = joblib.load('ism_avg_recall_list_3.pkl')

print("Plotting precision recall curves.")
plot_precision_recall(pm_avg_precision_list, pm_avg_recall_list, "popularity_model",
                      ism_avg_precision_list, ism_avg_recall_list, "item_similarity_model")
```

```
Plotting precision recall curves for a larger subset of data (12,000 rows) (user sample = 0.05).
Plotting precision recall curves.
```



Finally, the curve shows that the personalized model provides much better performance over the popularity model.

The collaborative which mean the personalized model use K-nearest Neighbor algorithms and it recommend based on some user's emotion or interest which is also work for users which will have the same emotion or interest. It also working for the songs behavioral, song which can have same behavior or song group will also recommend for users which are interested on.