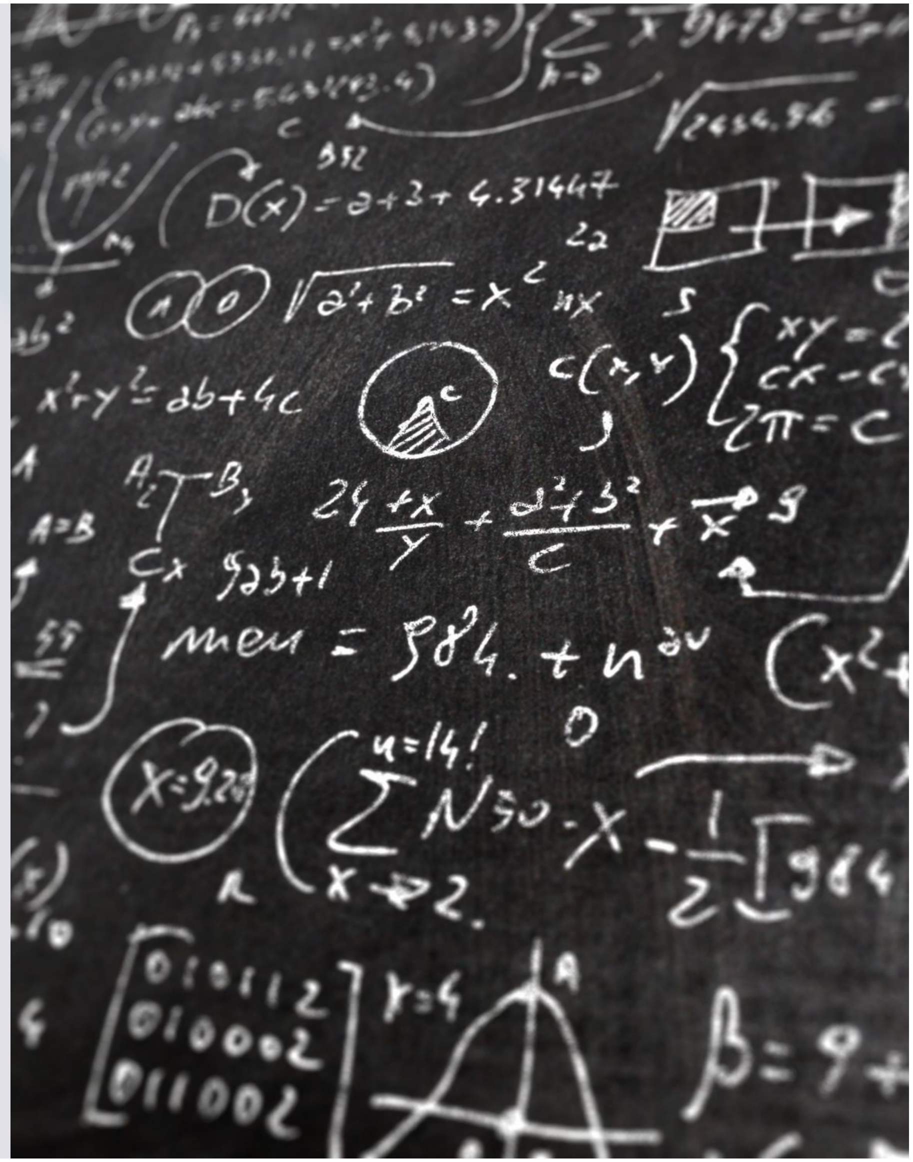
Music Recommendation System Using Implicit Feedback

Presented by Gaayathri Vaidhyanathan
Under the Guidance of Dr. Yingshu Li
Department of Computer Science
Georgia State University

OVERVIEW

- BACKGROUND
- PROBLEM FORMULATION
- OBJECTIVE
- METHODOLOGY OVERVIEW
- MODELING
- IMPLEMENTATION DEMO
- SIMULATION
- OBSERVATIONS AND RESULTS
- REFLECTIONS

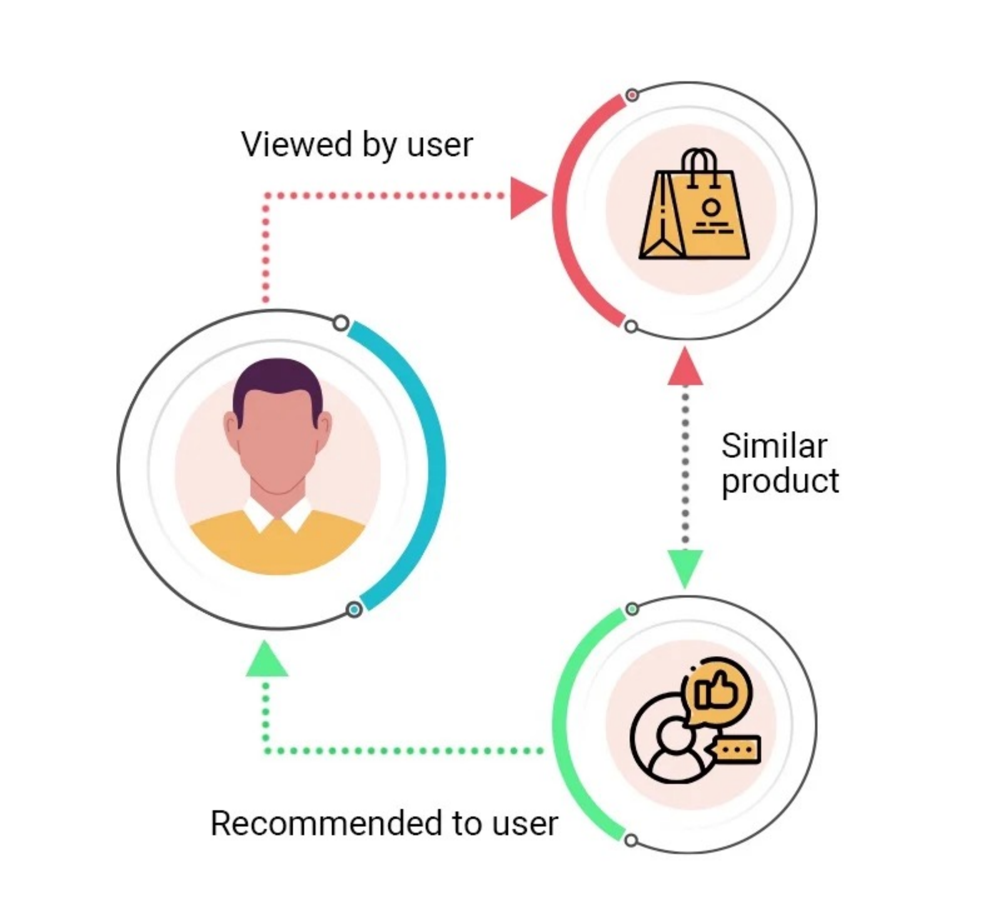


BACKGROUND

- Recommendation Engines try to predict the preferences of the clients and provide items that are likely to be appealing to them.
- A recommendation system can be built using a variety of methods, such as a popularity-based recommendation, a collaborative filtering approach (based on users and based on items), a content-based filtering approach, and a hybrid approach.

BACKGROUND

- The efficiency of the model can be evaluated based on the explicit feedback from the user ratings such as Mean Average Precision (MAP@K) and Mean Average Recall (MAR@K) and the recommendations keep improving based on the ratings received by the user.



PROBLEM FORMULATION

THE SPARSITY OF DATA AVAILABLE TO PRODUCE RECOMMENDATIONS: NOT ALL USERS/CUSTOMERS LISTEN TO ALL SONGS.

THE COMPUTATIONAL POWER OR ITERATIONS IT TAKES TO PRODUCE A RECOMMENDATION IS EXTENSIVE: NUMBER OF ITERATIONS = ΣALL_SONGS * ΣUSER_LISTENED_SONGS

THE EVALUATION METRICS: NOT FEASIBLE TO ACCUMULATE USERS FOR SURVEY/BETA TESTING.

OBJECTIVE

1

- Design a model to overcome sparsity in data when constructing similarity matrix.

2

- Decrease computational intensity/number of iterations

3

- Come up with an implicit feedback approach to evaluate the model based on data in hand.

METHODOLOGY OVERVIEW

Data Source - The dataset used for this project is the Million Song Dataset (MSD). Here, the core data focus is the Taste Profile Subset data provided by The Echo Nest.

Exploratory Data Analysis - A detailed analysis of the data is performed to have a better understanding of the data. Data is plotted to check for normal distribution and skewness/bias of data along with a probability plot to determine the measure of kurtosis.

Modeling - The algorithms developed are a popularity-based model, a similarity-based model constructing a cooccurrence matrix, matrix factorization-based approach using singular value decomposition.

Evaluation - Implicit feedback mechanism is designed and the measure used to compare these models is the cumulative probability of users listening to the recommendation provided.

BASELINE MODELS

Two baseline models are used to compare our proposed model:

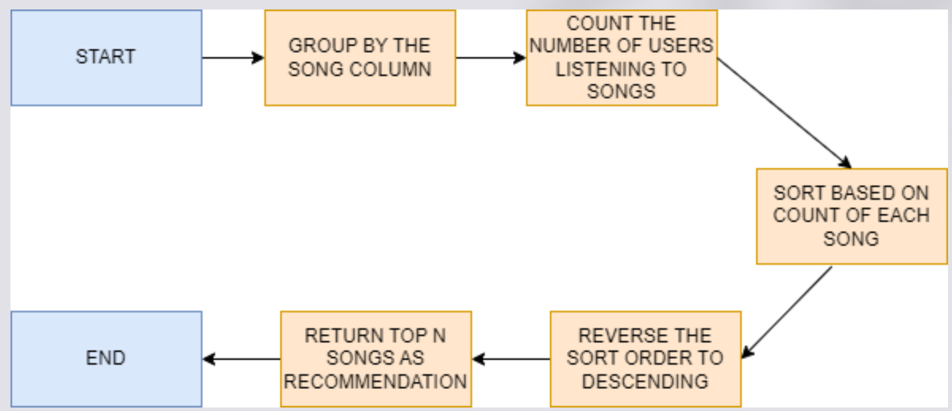
- Popularity-based Recommendation: Returns and introduces user to trendy music but lacks customization
- Traditional Collaborative Filtering i.e., Similarity-based Recommendation: Customizes the recommendation as per user's taste but is time-consuming

POPULARITY-BASED ENGINE

Algorithm

INPUT – Dataset
OUTPUT – Top 10 recommended songs
1: Load Dataset
2: Group by 'song' column
3: For each 'song':
4: Store count under each group as song_popularity
5: End For each
6: Sort based on song_popularity
7: Reverse the sorted data
8: Return Top 10 song details from data

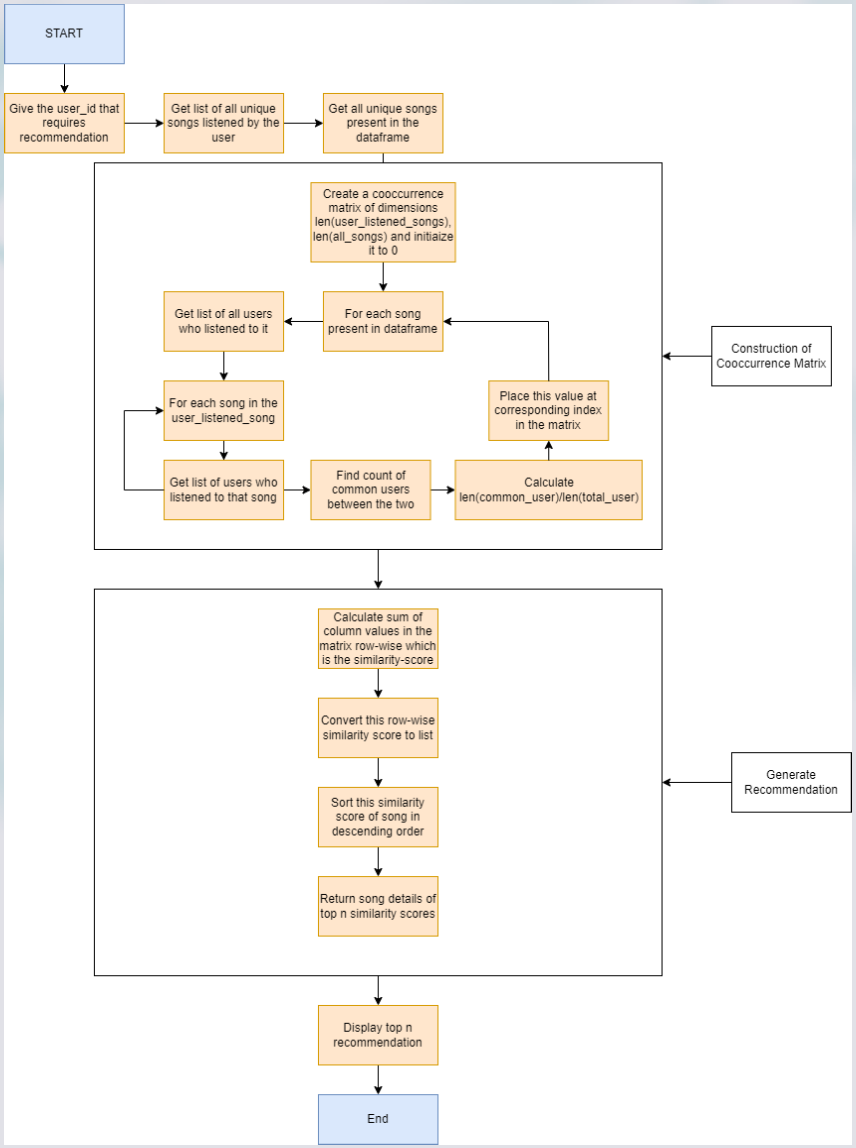
Flow of Algorithm



POPULARITY-BASED ENGINE

Code Snippet

```
# creating recommendation model based on popularity
def create_pre(main_df, user_id, song):
    song_popularity = main_df.groupby(['song'])['user_id'].count().reset_index()
    song_popularity.rename(columns = {'user_id': 'score'},inplace=True)
    # sorting it based on the popularity to provide top n recommendations based on popularity
    song_popularity_sorted = song_popularity.sort_values(['score', 'song'], ascending = [0,1])
    # adding rank into the sorted dataframe for better identification of top n recommendations
    song_popularity_sorted['rank'] = song_popularity_sorted['score'].rank(ascending=0, method='first')
    # if you want top n recommendations based on popularity. The value of n can be given as required by the user.
    recommendation_pre = song_popularity_sorted.head(10) # the number within head() can be changed based on the n value
    del recommendation_pre['score']
    return recommendation_pre
```



SIMILARITY-BASED ENGINE

SIMILARITY-BASED ENGINE

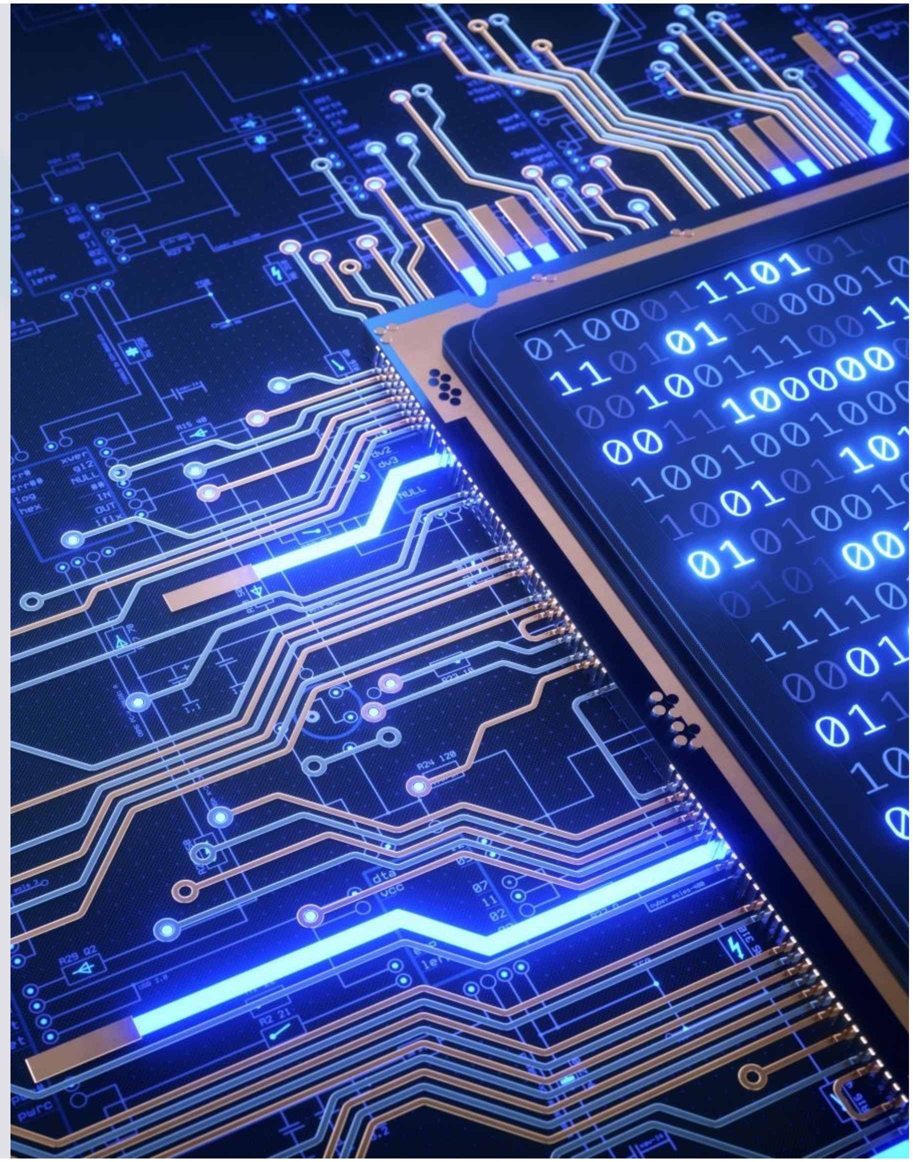
Whiteboard Explanation

PROPOSED MODEL

Matrix Factorization using Singular Value Decomposition:

- Matrix factorization decomposes matrix into constituents which when multiplied returns original matrix.
- Can be used to find Latent Factors between two entities.

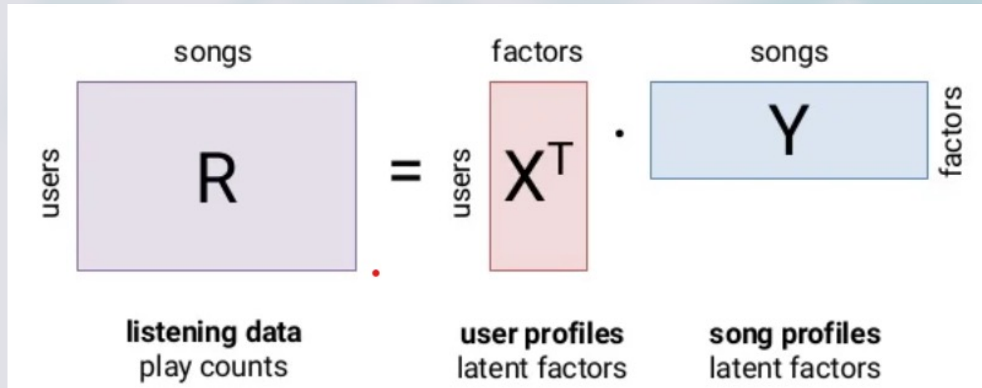
So, what are latent factors?



PROPOSED MODEL

Goal is to construct two matrices X and Y such that their product (matrix multiplication) roughly approximates R, assuming that the procedure assists in the identification of latent factors/features, denoted as K.

- X = |U| x K matrix (A matrix with dimensions of num_users * factors)
- Y = |P| x K matrix (A matrix with dimensions of factors * num_songs)



MATRIX FACTORIZATION-BASED ENGINE

Whiteboard Explanation

Singular Value Decomposition

Given some input matrix M, the formula for SVD can be outlined as seen below:

$$\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$$

M : An m x n matrix which you want to decompose

U : An m x m complex unitary matrix (left singular vectors)

Σ : An m x n rectangular diagonal matrix (holds the eigenvalues)

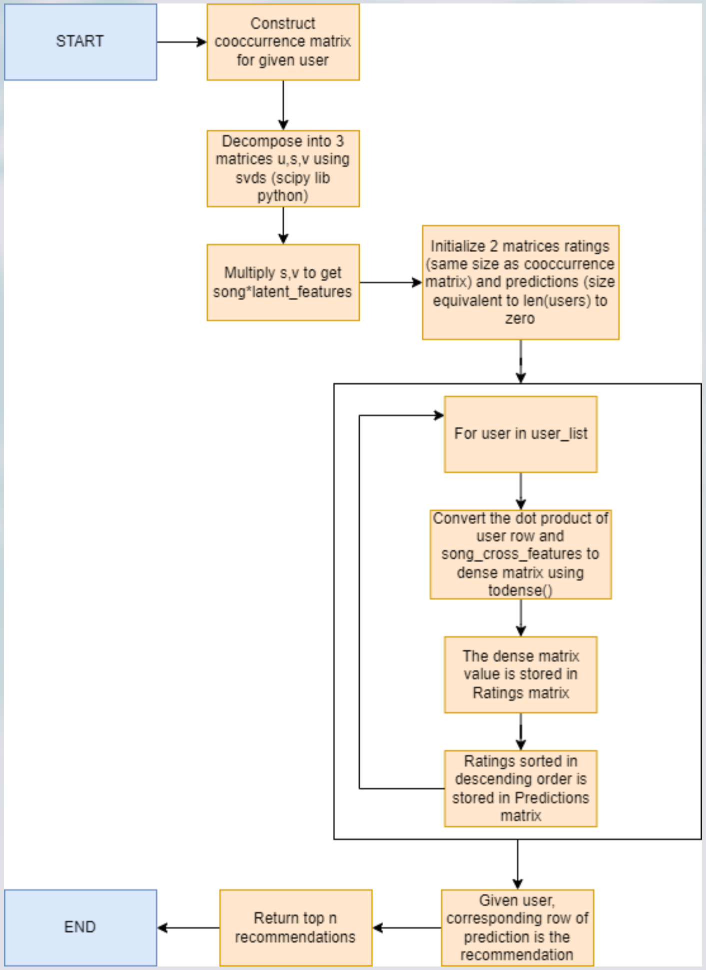
V : An n x n complex unitary matrix (right singular vectors)

Step 1: Transform the matrix M into a square matrix by multiplying it by its transpose: $M*M^T$

Step 2: Calculate the eigenvalues & eigenvectors of matrix $M*M^T$. The results of this will correspond to the Σ and U matrices.

Step 3: Solve for V by using the following formula: $V = 1/\Sigma * M^T * U$

MATRIX FACTORIZATION-BASED ENGINE



INPUT – Primary Matrix, No of Latent Factors(k)

OUTPUT – 3 Decomposed Matrices U, S, V

1: Use SVDs from Scipy Library

2: Decompose into 3 U, s, V

3: For each x in s:

4: S := sqrt(x)

5: End For each

6: Convert U, S, V into matrix using csc_matrix

7: Return U, S, V

INPUT – U, S, V, user_list

OUTPUT – Top 10 recommended songs

1: Multiply S, V and store as song_features

2: Initialize 2 matrices Ratings and Predictions to 0

3: For each user in user_list:

4: product := user * song_features

5: Rating[user] := convert product matrix to dense matrix

6: Predictions[user] := Sort Ratings

7: End For each

8: Return Top 10 from Predictions[user]

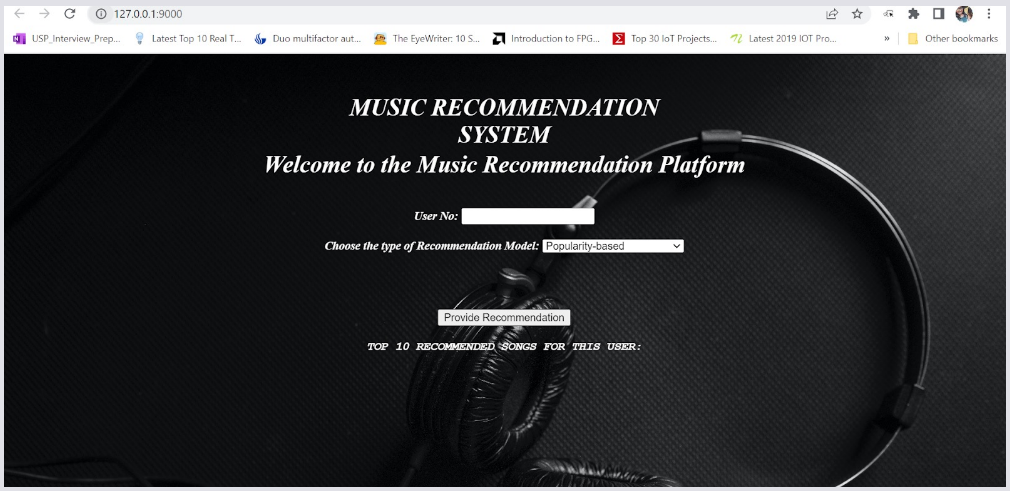
EVALUATION METRICS

INPUT – Dataset, Song
OUTPUT – Probability of the Song being listened to
1: Group the dataset by songs 2: Search for the input song group 3: Count the users who listened to the song 4: Probability = count/length(dataset)
5: Return Probability

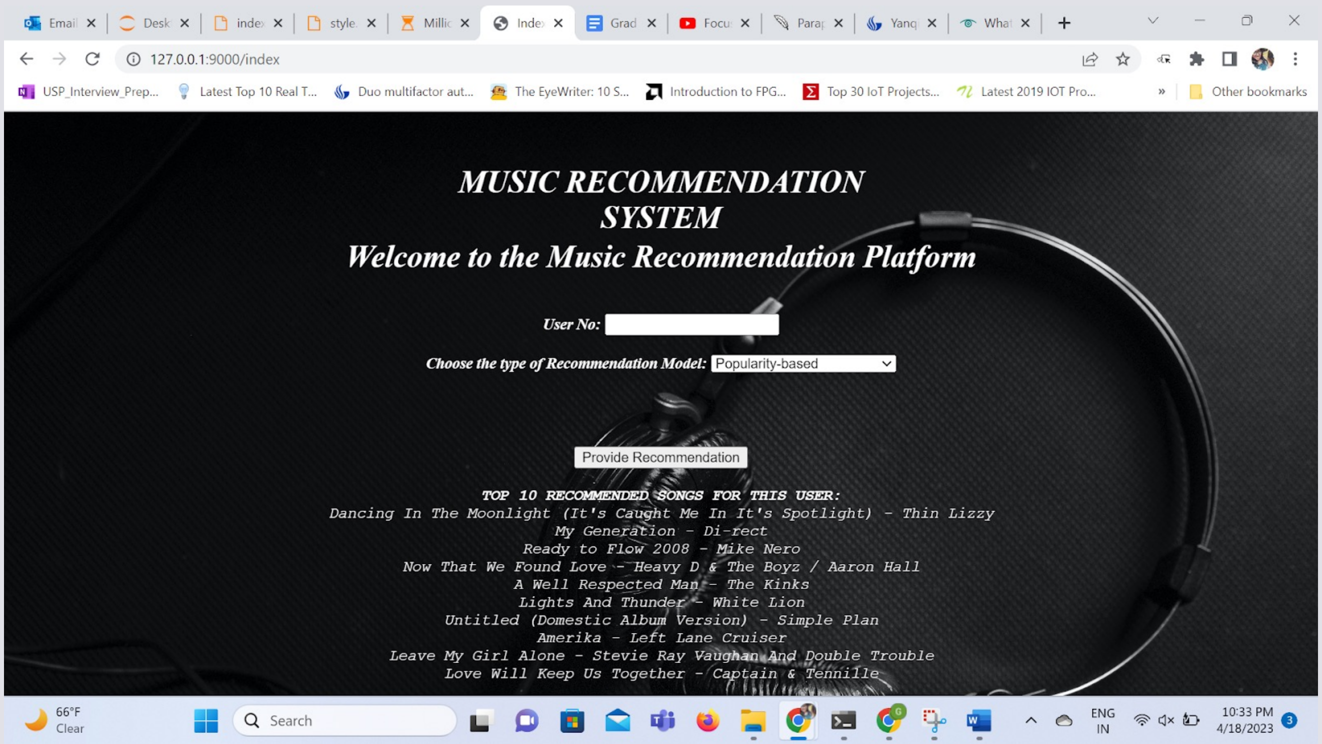
The cumulative probability of a given recommendation model can be determined by calculating the sum of all top n recommended songs that are the output of a given recommendation model.

IMPLEMENTATION

SIMULATIONS

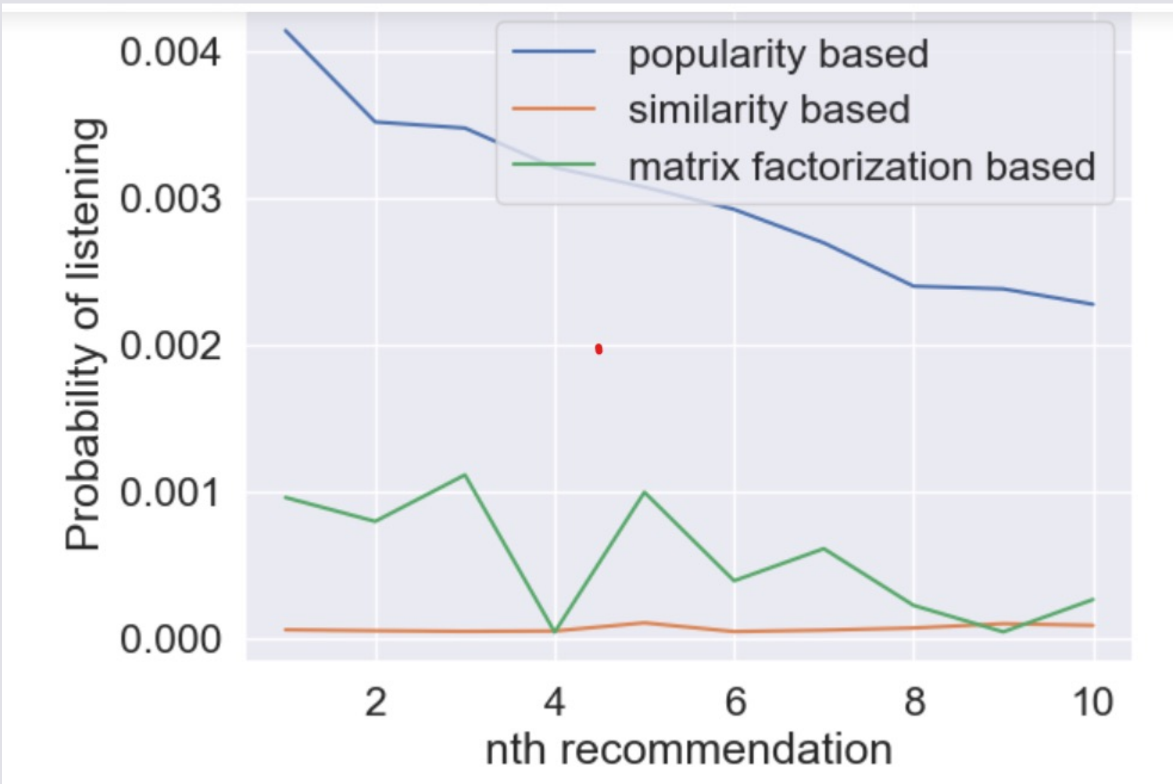


SIMULATIONS



OBSERVATIONS AND RESULTS

OBSERVATIONS AND RESULTS



REFLECTIONS

- Popularity-based: lacks customization
- Similarity-based: time-consuming
- Matrix factorization-based: quick and has customization

To be noted:

Possibility of overfitting in matrix factorization-based, so number of latent factors must be optimal

FUTURE SCOPE

- Come up with a weighted metrics based on positioning of the recommendation.
- Compare and contrast several matrix factorization techniques

REFERENCES

- Chen, Zhaoliang, and Shiping Wang. "A review on matrix completion for recommender systems." *Knowledge and Information Systems* (2022): 1-34.
- Rakshit, Pranati, Sougata Saha, Arindam Chatterjee, Subhayan Mistri, Swagata Das, and Gunjan Dhar. "A Popularity-Based Recommendation System Using Machine Learning." In *Machine Learning in Information and Communication Technology: Proceedings of ICICT 2021, SMIT*, pp. 143-150. Singapore: Springer Nature Singapore, 2022.
- Vineela, A., G. Lavanya Devi, Naresh Nelaturi, and G. Dasavatara Yadav. "A comprehensive study and evaluation of recommender systems." In *Microelectronics, Electromagnetics and Telecommunications: Proceedings of the Fifth ICMEET 2019*, pp. 45-53. Springer Singapore, 2021.
- Hu, Yifan, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets." In *2008 Eighth IEEE international conference on data mining*, pp. 263-272. Ieee, 2008.

THANK YOU!
ANY QUESTIONS/SUGGESTIONS?