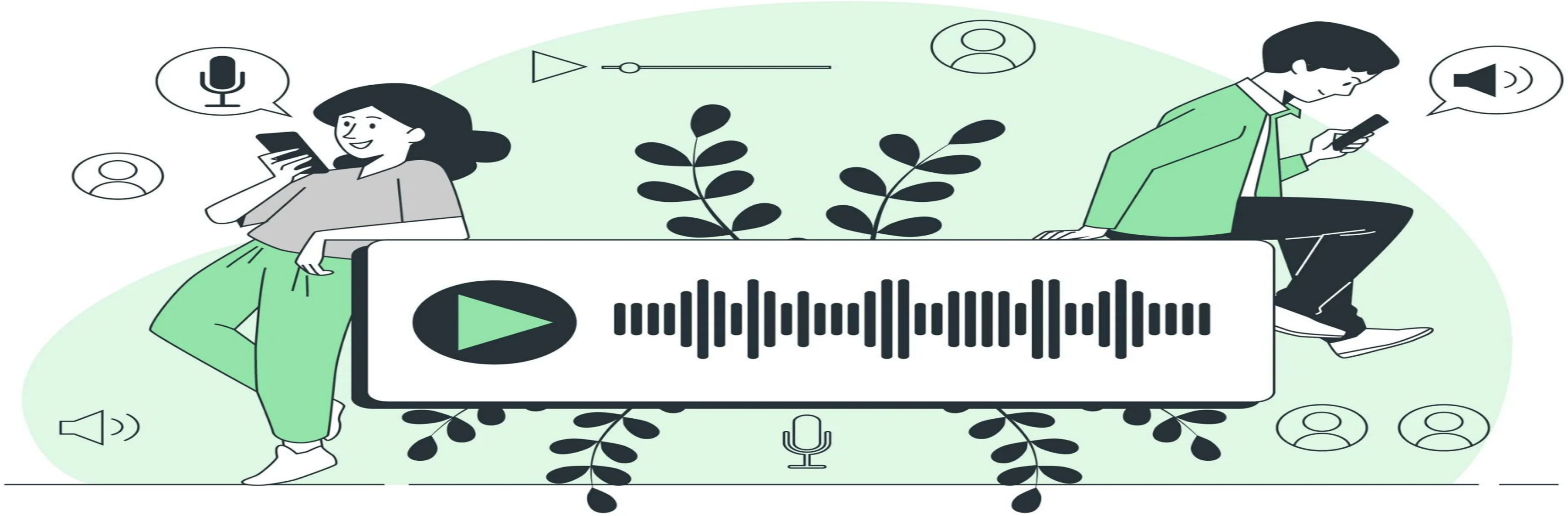
Music Recommender System

Presented by: Jayashree Lakshmi



Agenda

- ❑ Problem Definition
- ❑ Objective
- ❑ Solution Design
- ❑ Exploratory Data Analysis
- ❑ Performance Metrics
- ❑ Key observation and Insights
- ❑ Model Performance Overview
- ❑ Executive Summary

Problem Definition

Challenge: Difficulty in discovering new music that aligns well with our preferences

Current Limitation:

Manual search :Time consuming

Generic Recommendations: Lack of personalized suggestions

leading to unsatisfactory user experiences

Need: System which automatically recommends music based on the individual preferences and listening patterns

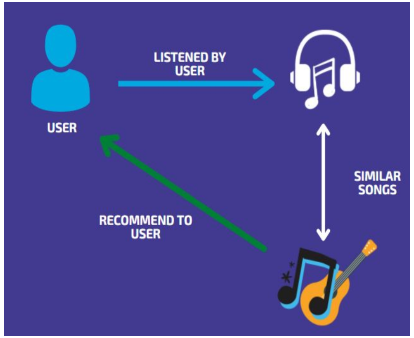
Objective

- To develop a personalized music recommender system that predicts the **Top N Songs** for each user.

  Key Features:

  User Data: Utilize user ID and play count to analyze listening behavior

  Song Data: Leverage song ID,song title and artist names to model song preferences.



Datasets

Dataset: Taste Profile Subset released by the Echo Nest as part of the Million Song Dataset

Song data: **1,000,000 Records**

song_id - A unique id given to every song

title - Title of the song

Release - Name of the released album

Artist_name - Name of the artist

year - Year of release

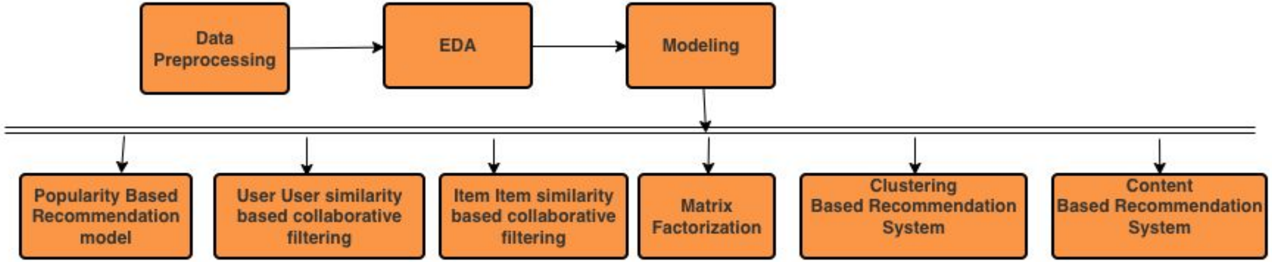
Count data: **2,000,000 Records**

user _id - A unique id given to the user

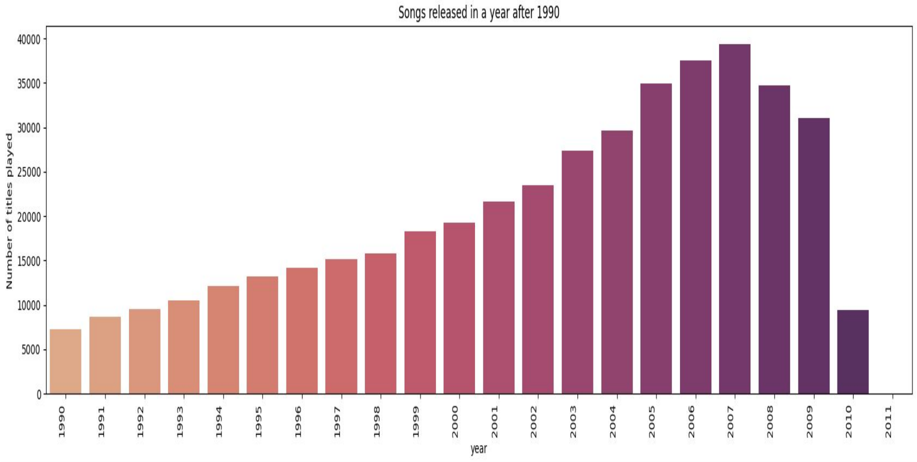
song_id - A unique id given to the song

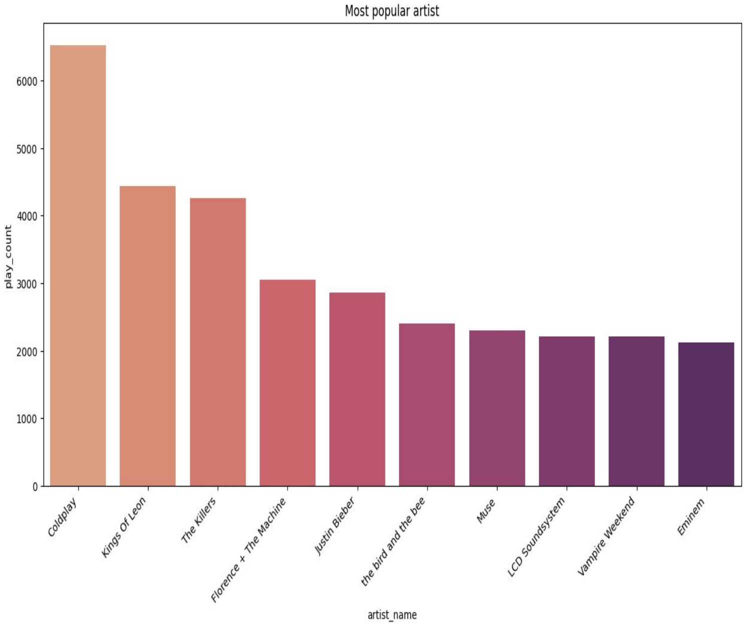
play_count - Number of times the song was played

Solution Design



Exploratory Data Analysis





Performance Metrics

Precision: Ratio of correctly predicted positive observations(True positive) to the total predicted positive observations(True positives + False Positives)

Imagine you have a playlist of songs you think your friend will like. Precision tells you how many of those songs your friend actually likes.

Recall: The ratio of correctly predicted positive observations (True Positives) to all observations in the actual class (True Positives + False Negatives).

How many of your friend's favorite songs you included in the playlist.

F1 score: The F1 score is the harmonic mean of Precision and Recall. It gives a single score that balances both precision and recall

Key Observations & Insights

Popularity Based Recommendation system:

Recommending popular/Trending items

Resolves :Cold Start Problem
Limitation:Lack of Personalization

User User Similarity Based Collaborative Filtering:

RMSE: 1.0817
Precision: 0.005
Recall: 0.003
F_1 score: 0.004

Tuning
Best parameters: {'k': 40, 'min_k': 10, 'sim_options': {'name': 'pearson_baseline', 'user_based': True}}
RMSE: 1.0131
Precision: 0.079
Recall: 0.038
F_1 score: 0.051
Tuning the User-User CF model significantly enhanced the recommendation quality,
resulting in more accurate, relevant, and balanced predictions.

Item Item Similarity Based Collaborative Filtering:

RMSE: 1.0320
Precision: 0.006
Recall: 0.007
F_1 score: 0.006

Tuning
Best parameters: {'k': 30, 'min_k': 3, 'sim_options': {'name': 'msd', 'user_based': False}}
RMSE: 0.8178
Precision: 0.028
Recall: 0.018
F_1 score: 0.022
Tuning the Item-Item Collaborative Filtering model resulted in improved accuracy and better-quality recommendations.

Matrix Factorization:

RMSE: 0.6590
Precision: 0.184
Recall: 0.103
F_1 score: 0.132

Tuning

Best parameters: {'n_epochs': 20, 'lr_all': 0.01, 'reg_all': 0.1}
RMSE: 0.7885
Precision: 0.074
Recall: 0.041
F_1 score: 0.053

The initial Matrix Factorization model performed better in terms of both precision, recall, and F1-score.

Content based Recommendation System:

```
[95] # Make the recommendation for the song with title 'Learn To Fly'
     recommendations('Learn To Fly', similar_songs)

     [445, 520, 246, 465, 367, 429, 0, 416, 417, 418]
     ['Big Me',
      'Everlong',
      'The Pretender',
      'Nothing Better (Album)',
      'From Left To Right',
      'Lifespan Of A Fly',
      'Daisy And Prudence',
      "Ghosts 'n' Stuff (Original Instrumental Mix)",
      'Closer',
      'No Cars Go']
```

Clustering based Recommendation System:

RMSE: 0.9698
Precision: 0.438
Recall: 0.653
F_1 score: 0.524

Tuning

RMSE: 0.9698
Precision: 0.439
Recall: 0.653
F_1 score: 0.525

Tuning the Cluster-Based Recommendation System resulted in minimal improvements in performance. The best parameters (n_cltr_u=3, n_cltr_i=5, n_epochs=40) slightly improved precision and F1-score, while maintaining the same recall and RMSE. This suggests that while the model is stable, further tuning might not result in significant gains.

Model Performance Overview

Model	RMSE	Precision	Recall	F1 Score	Content Score
User-User Collaborative Filtering	1.0131	0.079	0.038	0.051	N/A
Item-Item Collaborative Filtering	0.8178	0.028	0.018	0.022	N/A
Matrix Factorization (SVD)	0.6590	0.184	0.103	0.132	N/A
Cluster-Based Recommendation System	0.9698	0.439	0.653	0.525	N/A
Content-Based Recommendation	N/A	N/A	N/A	N/A	1.00
Hybrid Model (Example Prediction)	1.4063	N/A	N/A	N/A	N/A

Hybrid model's prediction error is relatively high compared to some of the individual models

Further **tuning** the hybrid model, potentially adjusting the weighting of the individual components

Executive Summary

Key Takeaways

- Fine-tuning hyperparameters improved performance metrics significantly, especially for Collaborative Filtering models
- Content-based and rank based filtering addresses the **cold start** problem, making it effective for recommending new or less-interacted songs
- Combines multiple techniques (Collaborative Filtering, Content-Based, Matrix Factorization) for better accuracy and diversity in recommendations.
- The hybrid approach can be extended to other domains like movies, books, and e-commerce recommendations.

Next Steps

- Experiment with deep learning models (e.g., neural collaborative filtering, autoencoders) to capture more complex user-item interactions.
- Implement strategies such as leveraging social media data or explicit user inputs to address the cold start problem for new users and items.
- Explore additional tuning and optimization techniques, especially for the hybrid model
- Develop a user-friendly front-end interface to deliver recommendations in a more engaging and intuitive way
- Conduct real-world testing with users to validate model performance and improve personalization through feedback.

```
[121] print ('!! THANK YOU !!')
!! THANK YOU !!
```