



Search



Home



Library



Our Music Playlists

Million Song Dataset Challenge: **Music** **Recommendation System**

Mary Erikson, Maya Zhao, Wanxuan Zhang, Xian Tang





Table of contents

01

Business
Problem



02

Executive
Summary



03

Exploratory
Data Analysis



04

Modeling &
Summary



05

Methodology &
Model Evaluation



06

Findings &
Conclusions



Business Problem

Recommendation system is widely utilized by companies such as Amazon, Google, and Spotify to increase user interaction and enrich business potential.

Our Objective: Building a state-of-the-art recommendation system for the streaming music industry



Improve Recommendation Performance



Increase Customer Retention



Achieve Business Success

Executive Summary

We built these recommendation models in a three-pronged approach:

1

Data Prep and Analysis

- Clean and create curated datasets
- Check for outliers/data validity
- Visualize distribution of all the variables
- Prep data for modeling

2

Model Creation

- Created a **baseline** model (song popularity)
- Created 3 personalized models to compare it to
 1. **Content-based model**
 2. **Collaborative filtering item-item based model**
 3. **Collaborative filtering item-user based model**

3

Model Comparison

- Since accuracy metrics weren't the best evaluation fit for our use case, **we ran each model for one user** in order to **directly compare the recommendations**
- We then evaluated **the best use case for each recommendation** output

Data Description

Dataset: Million Song Dataset Challenge (by Kaggle)

MSD Full dataset:

- **1,019,318** unique users
- **384,546** unique MSD songs
- **48,373,586** user - song - play count triplets

The core data for our project is the [Taste Profile Subset](#) released by [The Echo Nest](#) as part of the Million Song Dataset. It consists of:

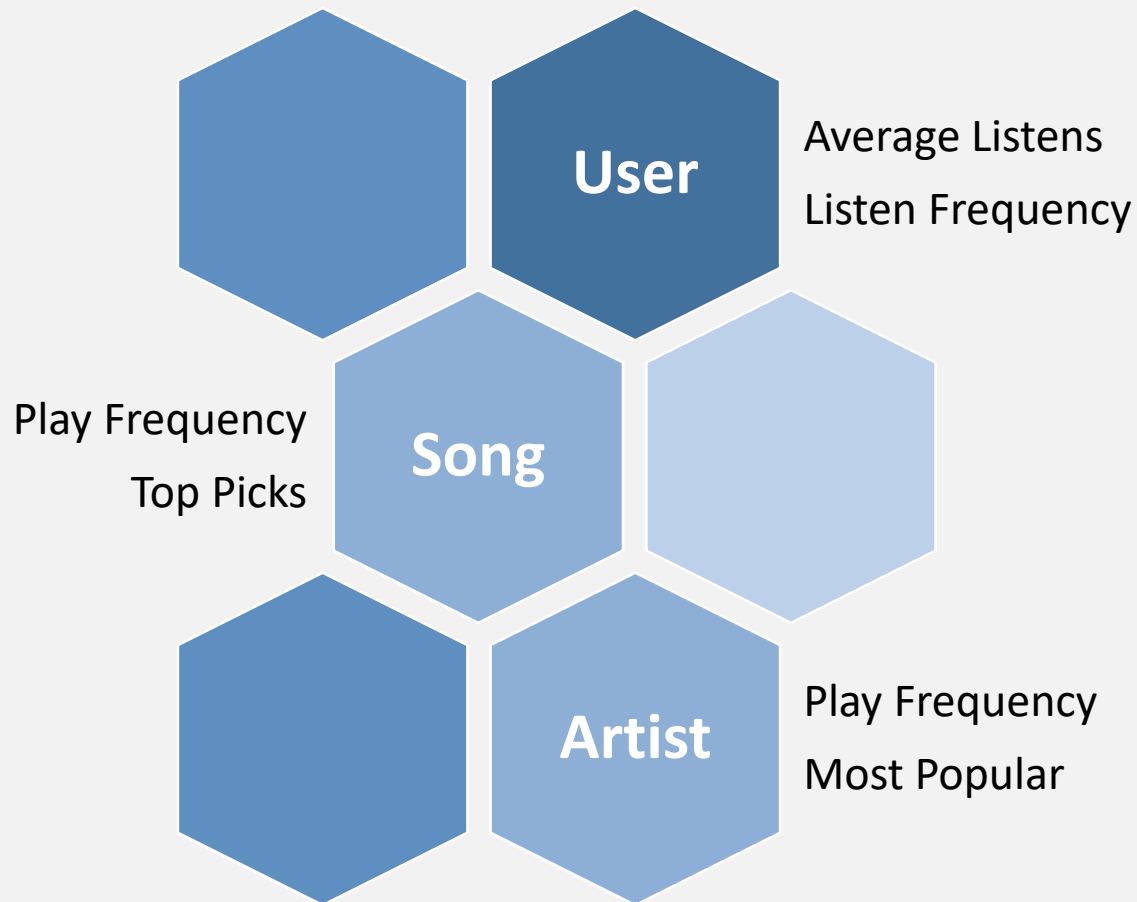
- triplets: 'kaggle_visible_evaluation_triplets.txt' (user ID, song ID, play count)
- mapping from song to tracks: 'taste_profile_song_to_tracks.txt'
- mapping from tracks to artists: 'unique_tracks.txt'

Additionally, we included the following datasets for the content-based model:

- Location of artists: 'artisit_location.txt'
- Year of tracks: 'tracks_per_year.txt'
- Genre of songs: 'song_genre.cls'



EDA - Overview



Song

- Unique songs: 386,213
- Unique songs listened by users: 163,026
- Unique genre: 15
- Year: 1922 - 2011

Song-to-tracks

- 1-1 mapping: 383,164
- 1-2 mapping: 2,380
- 1-3 mapping: 39



User

- Unique user: 110,000

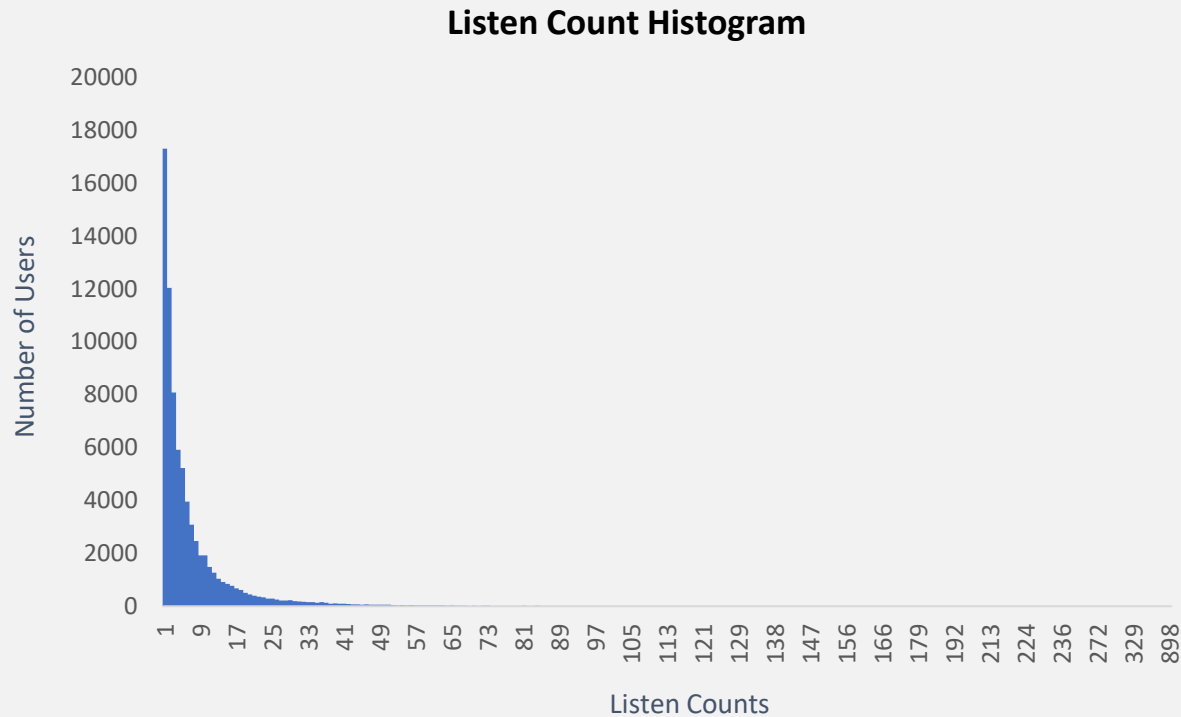


Artist

- Unique artist: 72,665
- Unique tracks: 1,000,000
- Unique location: 3,989 (for 13,629 artists)

EDA - User

- **Right-skewed** distribution of user listen count
- Most users' listen count are less than **50**
- Users listen to favorite songs frequently



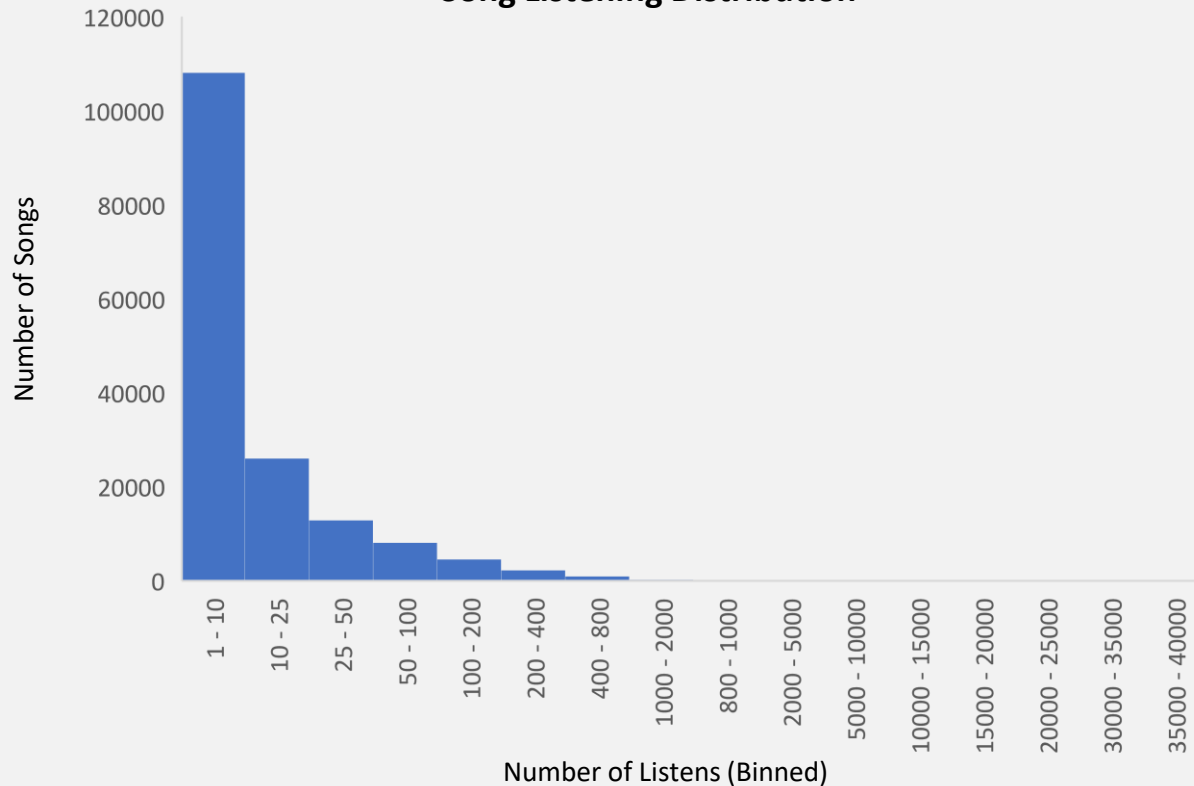
Sample User Listening Frequency

Title	Artist	Listen Count
Load Murmuring	Octopus Project	10
Just A Man	Faith No More	10
Fast Car	Tracy Chapman	6
Don't Worry Be Happy	Bobby McFerrin	2
Hands To Heaven	Breathe	2
Stop	Sam Brown	2
Este amor	Federico Aubele	1
Undead	Adagio	1
Picture U & Me	Mo B. Dick	1
Your Love Is Better Than Life	Newsboys	1

EDA - Song

- Most songs are played between 1-10 times (66%)
- Number of times a song is played is very heavily **right skewed**

Song Listening Distribution

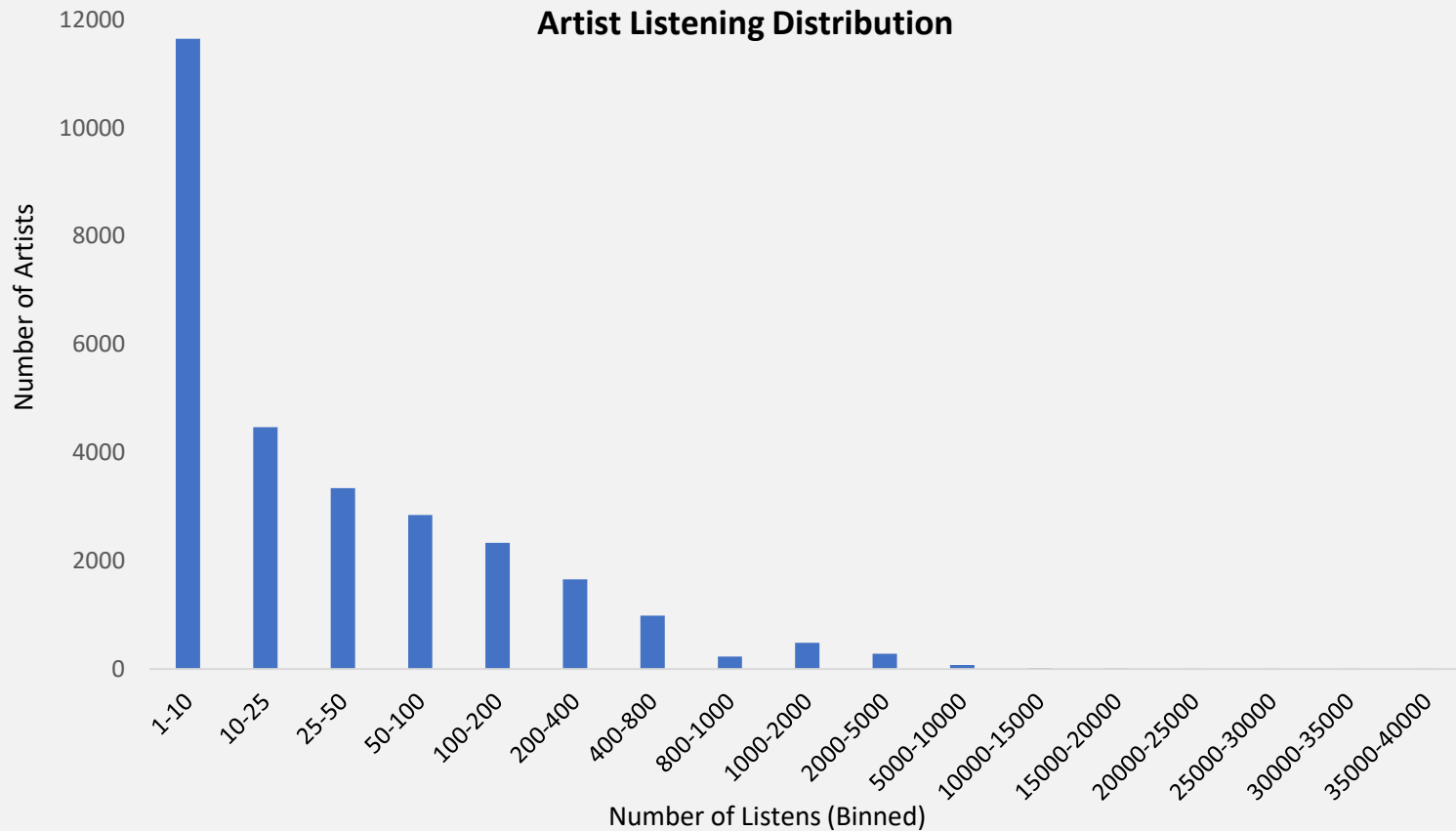


Top 10 Songs by Number of Listens

Title	Artist Name	Listen Count
You're The One	Dwight Yoakam	35,432
Undo	Björk	33,179
Revelry	Kings Of Leon	24,359
Sehr kosmisch	Harmonia	19,454
Horn Concerto No. 4 in E flat K495: II. Romance (Andante cantabile)	Barry Tuckwell/Academy of St Martin-in-the-Fields/Sir Neville Marriner	17,115
Dog Days Are Over (Radio Edit)	Florence + The Machine	14,279
Secrets	OneRepublic	12,392
Ain't Misbehavin	Sam Cooke	11,610
Invalid	Tub Ring	10,794

EDA - Artist

- Most artists are played between 1-10 times (41%)
- Frequencies of an artist played are very heavily **right skewed**

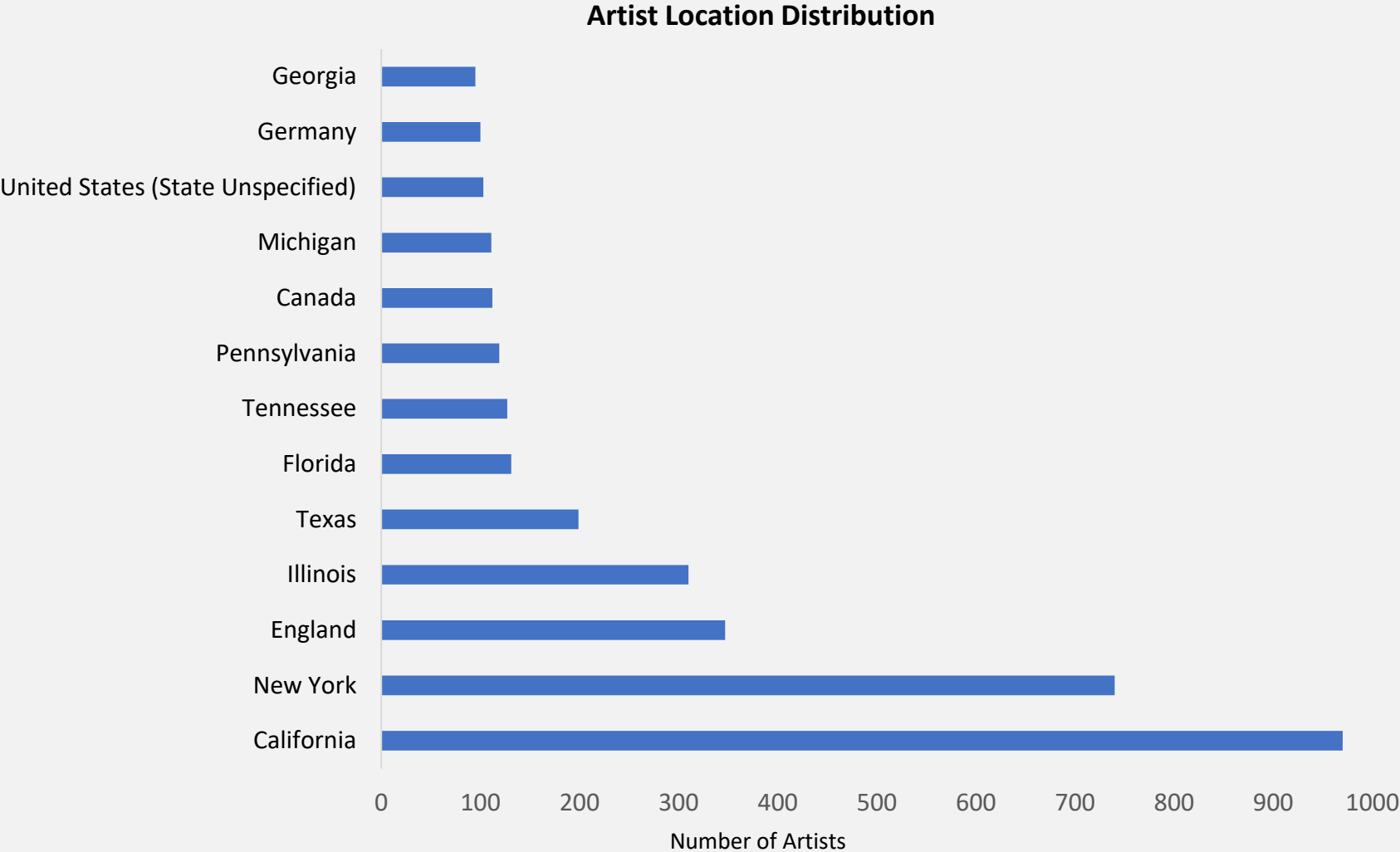


Top 10 Artists by Number of Listens

Artist Name	Listen Count
Kings Of Leon	35,857
Dwight Yoakam	35,688
Björk	35,210
Coldplay	32,135
Florence + The Machine	28,224
Justin Bieber	26,133
Alliance Ethnik	21,603
Train	21,356
OneRepublic	20,802
Harmonia	19,461

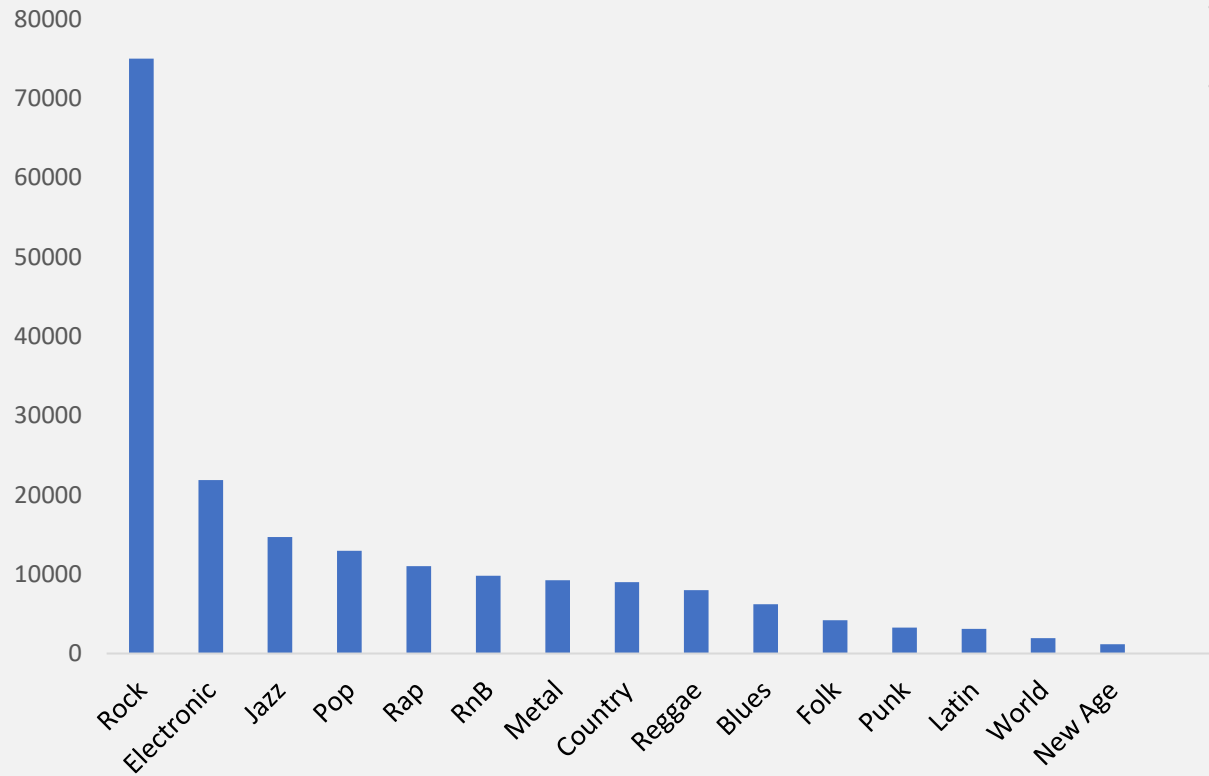


EDA – Artist Location

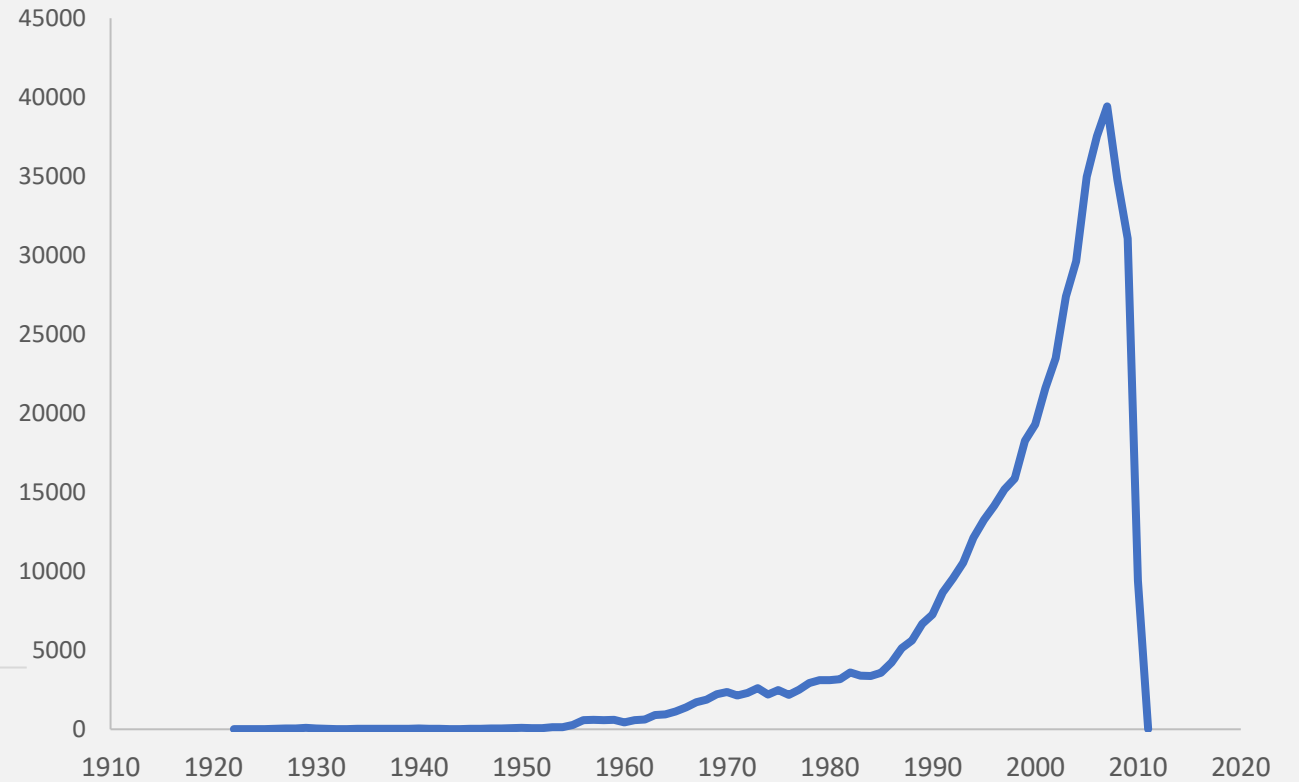


EDA – Genre & Year

Song Genre Distribution



Number of Songs by Year





Modeling Overview

01 Baseline Model

Our most simplistic model, recommending the most popular songs. Neither user nor content specific.



02 Content Based

This model utilizes what the user historically likes to make a recommendation of similar songs they haven't listened to yet.



03 CF Item-Item Based

This model identifies what songs a user has liked in the past, and what other songs that user would like based on the similarity other songs have to the user's current taste.



04 CF User-Item Based

This model identifies what songs a user has liked in the past, and what other songs that user would like based on what other similar users have liked.



Single User Recommendation Comparison

We then ran each of these models on **user "fd50c4007b68a3737fe052d5a4f78ce8aa117f3d"** and compared the recommendations. We then concluded the proper use case for each model based on what a given user may be looking for in their recommendation engine.

Baseline Model – Based on Popularity Only

Key Idea: get a list of songs only based on their popularity (play_count)

Step 1: Data Transformation

- Concatenate song title with artist name
- Group by 'song_artist' and aggregate by 'play_count'
- Calculate percentage using single song 'play_count'/total listens

Step 3: Train & Split

- Split the train and test dataset with 7:3 ratio

Step 2: Build Model (key part)

- Generate a score for each unique songs based on user_ids
- Sort the songs based on score
- Come out with recommendation rank based on score
- Get top 10 recommendations

Step 4: Run the model with a test user_id

Sample Output (would be the same for all user id as it's solely focused on popularity)

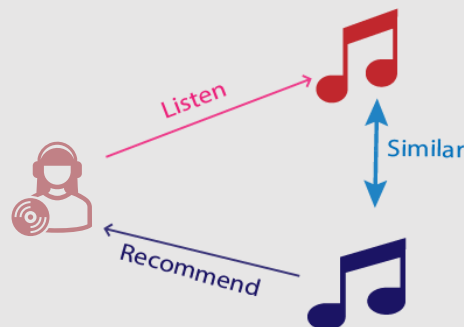
	user_id	song_artist	score	Rank
100543	91b8fac7dc5e03f6cfaf6e2aa7171f14a8354d62	Sehr kosmisch - Harmonia	3513	1.0
127831	91b8fac7dc5e03f6cfaf6e2aa7171f14a8354d62	Undo - Björk	3168	2.0
138725	91b8fac7dc5e03f6cfaf6e2aa7171f14a8354d62	You're The One - Dwight Yoakam	2894	3.0
28847	91b8fac7dc5e03f6cfaf6e2aa7171f14a8354d62	Dog Days Are Over (Radio Edit) - Florence + Th...	2640	4.0
95557	91b8fac7dc5e03f6cfaf6e2aa7171f14a8354d62	Revelry - Kings Of Leon	2580	5.0
100324	91b8fac7dc5e03f6cfaf6e2aa7171f14a8354d62	Secrets - OneRepublic	2397	6.0
50011	91b8fac7dc5e03f6cfaf6e2aa7171f14a8354d62	Horn Concerto No. 4 in E flat K495: II. Romanc...	2264	7.0
48476	91b8fac7dc5e03f6cfaf6e2aa7171f14a8354d62	Hey_ Soul Sister - Train	1956	8.0
37981	91b8fac7dc5e03f6cfaf6e2aa7171f14a8354d62	Fireflies - Charttraxx Karaoke	1912	9.0
123715	91b8fac7dc5e03f6cfaf6e2aa7171f14a8354d62	Tive Sim - Cartola	1877	10.0



Content-based Model

Key Idea:

predict what a user like based on what that user like in the past



Example:

recommend music for someone who listens to 2000s rock music

	title	artist_name	genre	location	year	distance	rank
	Know Too Much	Turbo Fruits	Rock	Tennessee	2007.0	0.000258	1.0
	Ramblin Rose	Turbo Fruits	Rock	Tennessee	2007.0	0.000258	2.0
	Waiting For The Rain To Come	Griffin House	Rock	Nashville, TN	2007.0	0.000795	3.0
	Black Thumbnail	Kings Of Leon	Rock	Nashville, Tennessee	2007.0	0.000795	4.0
	Arizona	Kings Of Leon	Rock	Nashville, Tennessee	2007.0	0.000795	5.0

Algorithms:

1. Data Transformation:

- Normalize genre, year, and artist location features using *StandardScaler*.
- Compute the average vector of the features for each song the user has listened to.

2. Compute Distance:

Find the n -closest data points in the dataset (excluding the points from the songs in the user's listening history) to this average vector.

3. Recommend:

Take these n points and recommend the songs corresponding to them.

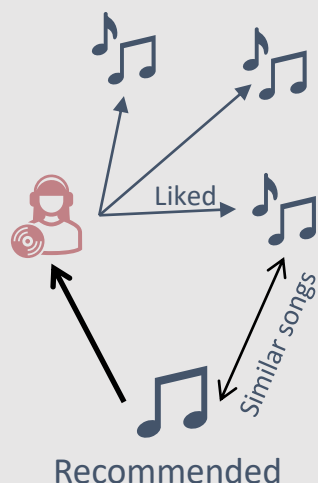
Distance metric between two data points: cosine distance

$$distance(u, v) = 1 - \frac{u \cdot v}{\|u\| \|v\|} = 1 - \cos \theta$$

Collaborative Filtering – Item-Item Based Model

Key Idea:

Looks for the items the user has previously consumed and then finds other similar items yet to be consumed and recommends accordingly.



Pro: Usually more accurate/personalized than user-based models.

Con: Takes a long time to run because it calculates concurrence matrix of all user songs vs. all songs in dataset, then calculate similarity "score" for recommendation.

Algorithms:

1. Feature Engineering:

- Lack the concept of "rating".
- Transformed play counts to a fractional number representing its fraction in each user's total count of plays, measuring the degree of "likeness" for a song in the [0,1] range.

2. Matrix factorization :

- Build a sparse concurrence matrix, a user's listened to songs versus all songs in training set dimension matrix.
- Find similarity between all pairs of items by using cosign of rating vectors.
- Similarity function uses normalized ratings (each user's average rating).

3. Recommendation:

- Uses the most similar items to a user's already "rated" items to generate a list of recommendations using a weighted sum score.

$$\text{score}(u, i) = \frac{\sum_j^I \text{similarity}(i, j)(r_{(u,j)} - \bar{r}_j)}{\sum_j^I \text{similarity}(i, j)} + \bar{r}_i$$

Annotations for the formula:

- $\sum_j^I \text{similarity}(i, j)$: sum of item 'i' and 'j' similarity score
- $\text{similarity}(i, j)(r_{(u,j)} - \bar{r}_j)$: sum the multiplication of item 'i' and 'j's similarity and difference of rating by user 'u' to item 'j' and it's average rating
- $r_{(u,j)} - \bar{r}_j$: subtracting with average rating to normalize the rating scale
- \bar{r}_i : sum with user 'i's average rating

Collaborative Filtering – User-Item Based Model

Key Idea:

Taking user as the central entity, the algorithm will look for similarities among users and generate recommendations based on similarities between users.



Pro:

- Context independent
- More accurate than non-customized recommendations

Con:

- Cold start for new users who have no to little information
- New item: lack of ratings to create a solid ranking

Algorithms:

1. Feature Engineering:

- Lack the concept of "rating".
- Transformed play counts to a fractional number representing its fraction in each user's total count of plays, measuring the degree of "likeness" for a song in the $[0,1]$ range.
- Sort user_id and build a table of user index to ease the calculation process.

2. Matrix factorization :

- Build a sparse utility matrix, a user Vs item dimension matrix.
- Use the svds function provided by the scipy library to break down the utility matrix into three different matrices.
- Multiply the decomposed matrices provided by SVD and make candidate recommendations.

3. Recommendation:

- Sort and display recommendations for users

Model Comparison – Specific User Use Case

We generate recommendations using each model for the specific user with user_id: fd50c4007b68a3737fe052d5a4f78ce8aa117f3d, and here is the results:

Baseline Model: 34ms

	song_artist	score	Rank
	Sehr kosmisch - Harmonia	3513	1.0
	Undo - Björk	3168	2.0
	You're The One - Dwight Yoakam	2894	3.0
	Dog Days Are Over (Radio Edit) - Florence + Th...	2640	4.0
	Revelry - Kings Of Leon	2580	5.0
	Secrets - OneRepublic	2397	6.0
	Horn Concerto No. 4 in E flat K495: II. Romanc...	2264	7.0
	Hey_ Soul Sister - Train	1956	8.0
	Fireflies - Charttraxx Karaoke	1912	9.0
	Tive Sim - Cartola	1877	10.0

Content-based Model: 105ms

	title	artist_name	distance	rank
	Never Ending Song Of Love	Crystal Gayle	0.000052	1.0
	I'm in Love Again	George Morgan	0.001660	2.0
	Western Girls	Marty Stuart	0.001849	3.0
	Hillbilly Rock	Marty Stuart	0.001849	4.0
	Rub It In	Billy Crash Craddock	0.001913	5.0
	Statue Of A Fool	Jack Greene	0.001977	6.0
	Don't Close Your Eyes	Keith Whitley	0.003293	7.0
	I'm No Stranger To The Rain	Keith Whitley	0.003293	8.0
	Watch Me Fall	Uncle Tupelo	0.003305	9.0
	No Substitute	John Lee Hooker	0.003448	10.0

Model Comparison – Specific User Use Case

We generate recommendations using each model for the specific user with user_id: fd50c4007b68a3737fe052d5a4f78ce8aa117f3d, and here is the results:

Collaborative Filtering – User-Item Based Model: 8.59 secs

The number 1 recommended song is Check Mate BY Upside Down Room
The number 2 recommended song is Rianna BY Fisher
The number 3 recommended song is Cuenta Conmigo BY Jerry Rivera
The number 4 recommended song is It Takes A Fool To Remain Sane BY The Ark
The number 5 recommended song is Repair Machines BY Vitalic
The number 6 recommended song is Robot Soul (Radio Edit) BY Cosmo Vitelli
The number 7 recommended song is I Think I'll Live BY Charlie Louvin
The number 8 recommended song is Word Up! BY Cameo
The number 9 recommended song is Monster BY Lady GaGa
The number 10 recommended song is Fallen Angel BY Elbow

Collaborative Filtering – Item-Item Based Model: 1hr 23 mins 3 secs

	user_id	song	score	rank
0	fd50c4007b68a3737fe052d5a4f78ce8aa117f3d	Undo	0.047096	1
1	fd50c4007b68a3737fe052d5a4f78ce8aa117f3d	Revelry	0.040056	2
2	fd50c4007b68a3737fe052d5a4f78ce8aa117f3d	When You Feel The Mess	0.033967	3
3	fd50c4007b68a3737fe052d5a4f78ce8aa117f3d	I CAN'T GET STARTED	0.032593	4
4	fd50c4007b68a3737fe052d5a4f78ce8aa117f3d	Tive Sim	0.031691	5
5	fd50c4007b68a3737fe052d5a4f78ce8aa117f3d	Représente	0.030288	6
6	fd50c4007b68a3737fe052d5a4f78ce8aa117f3d	Ain't Misbehavin	0.029412	7
7	fd50c4007b68a3737fe052d5a4f78ce8aa117f3d	Invalid	0.028993	8
8	fd50c4007b68a3737fe052d5a4f78ce8aa117f3d	Sincerité Et Jalousie	0.025264	9
9	fd50c4007b68a3737fe052d5a4f78ce8aa117f3d	Canada	0.025001	10

Methodology

Tooling:

Jupyter Notebook & Google Colab



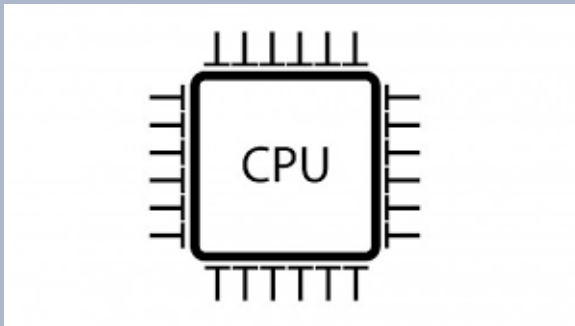
Deployment (in the future):

- Using Python and PySpark on Google Cloud Platform
- Embed the recommender API into mobile APPs



Hardware:

CPU



Data Update (in the future):










Once a month, using the data from active users



Findings and Conclusions



Recommendations

Use Case	Baseline Model	Content Based	FCUB	FCIB
New User				
New Song				
Active User				
Returning User				

- Baseline model compares popularity among all songs
- Content-based model compares similarities among all songs
- FCUB finds similar users and make recommendations
- FCIB finds similar songs and make recommendations

Assumptions

- Users who have similar preferences in the past are likely to have similar preferences in the future.

Impact

- Improve recommendation efficiency
- Increase user retention rate
- Achieve business success

Applications:

- Music apps / websites such as Spotify, Apple Music, etc.

Possible Extension:

- Movie recommendation
- Book recommendation
- ...
- Any scenarios involved recommendation with respective modification.

Lessons Learned and Recommendations

Next Steps:

- Use the original Million Song Dataset to get others features and apply
- Extend to cases where the dataset has “hating” score for songs
- Find different ways of measuring the "implicit feedback"

Additional Datasets:

- Lyrics of songs
- Audio files
- User profiles

Thoughts on Explainable AI:

- Comparing to some other black-box models, reasoning behind recommendation system is more intuitive
- Recommendation systems based on different algorithms are understandable: popularity based, content based, user similarity based, item similarity based...

References

- <http://millionsongdataset.com/tasteprofile/>
- <https://www.kaggle.com/competitions/msdchallenge/data>
- <https://www.kaggle.com/code/mgmarques/million-song-recommendation-engines>
- https://www.tagtraum.com/msd_genre_datasets.html
- <https://towardsdatascience.com/how-to-build-a-simple-song-recommender-296fcbc8c85>
- <https://towardsdatascience.com/how-to-build-an-amazing-music-recommendation-system-4cce2719a572>
- <https://docs.microsoft.com/en-us/archive/blogs/carlnol/co-occurrence-approach-to-an-item-based-recommender>
- <https://realpython.com/build-recommendation-engine-collaborative-filtering/>
- <https://towardsdatascience.com/comprehensive-guide-on-item-based-recommendation-systems-d67e40e2b75d>



Search



Home



Library



Our Music Playlists

Thanks for this amazing quarter!

Our GitHub Repository link:

<https://github.com/Maggie0927/MillionSongRecommendation>

