RECOMMENDER SYSTEM

Ivette M Tapia

MOTIVATION & DESCRIPTION

- **Business Case:** The Great Music Service company has a 1M catalog of songs to listen to available to their users. Given the large catalog of songs, no one user can be reasonably expected to go through all the content and choose what they want to listen to. Thus, ability to show users content that is relevant to them encourages greater engagement.
- **Variables:** user plays, artist, artist_familiarity, artist_hotttnesss, artist_name, release, song_hotttnesss, song_id, title, danceability, duration, energy, key, loudness, mode, tempo, time_signature, track_id, year, artists tags

WHAT ARE RECOMMENDATION SYSTEMS?

- Recommendation systems are a type of retrieval information technology.
- Recommendations engines differ from search in that:
  - Users will not necessarily tell the engine what they are looking for, or
  - Users may not know what they are looking for. These types of engine use individual’s explicit preferences or behaviors (implicit preferences) to retrieve information that is relevant to the specific active user.

GIT HUB REPOSITORY

Data & Exploratory Analysis Code

- Step 1: Data Wrangling
- Step 2: Preliminary Exploratory Analysis
- Step 3: Exploratory Analysis and Statistical Inference

Recommender Models

- Recommender Models

Final Report

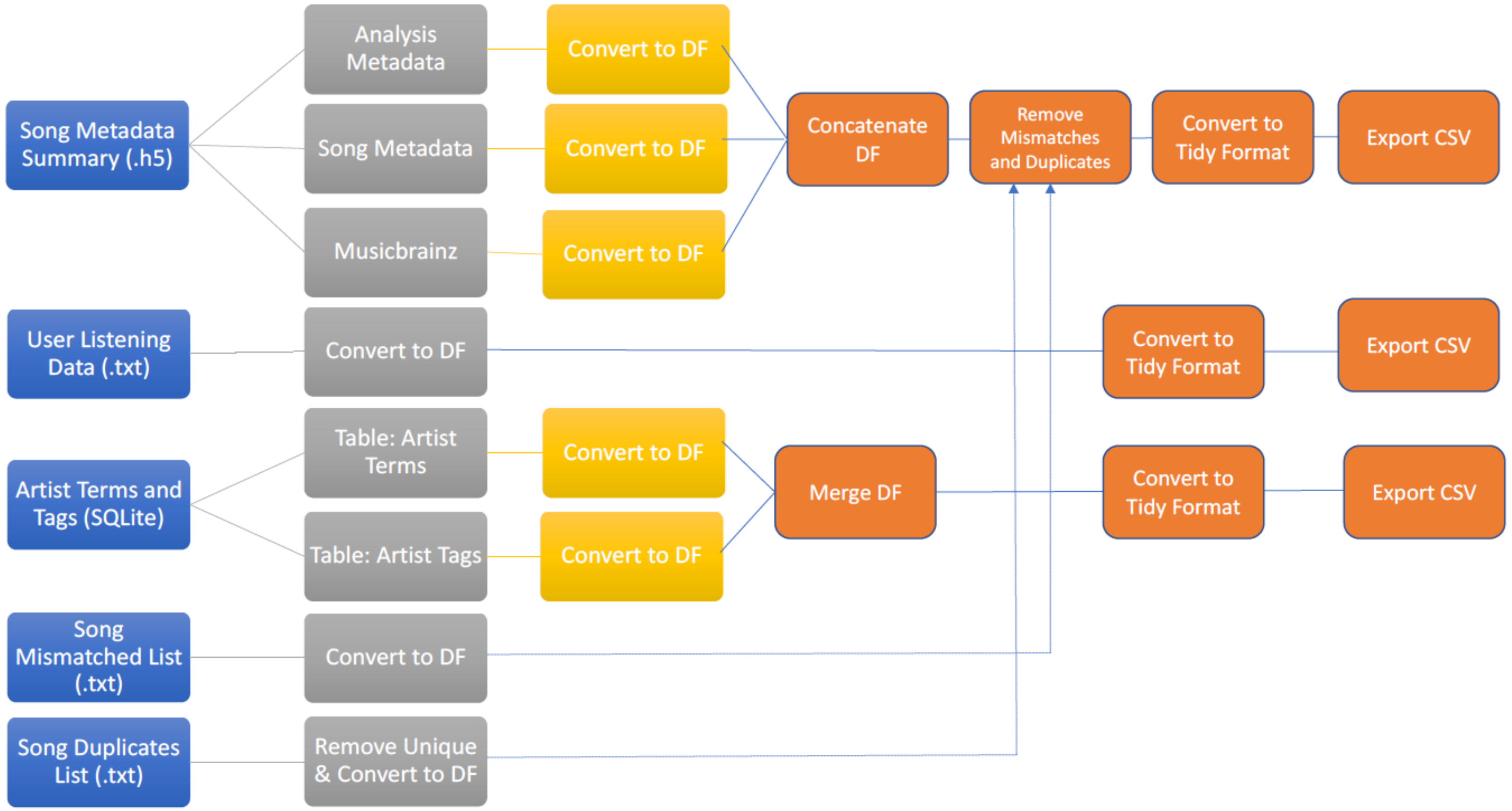
- Final Report

DATA ACQUISITION

COLUMBIA UNIVERSITY LABROSA

File	Description	Source	Type	Quantity of Files
Million Song Dataset	Collection of 1M .h5 files. Each file represents data for one song.	Using the terminal: rsync -avzuP publicdata.opensciencedatacloud.org::ark:/31807/osdc-c1c763e4/remotefile /path/to/local_copy	.h5	1M
Million Song Summary File	Song metadata information.	http://labrosa.ee.columbia.edu/millionsong/sites/default/files/AdditionalFiles/msd_summary_file.h5	.h5	1
Artist Terms and Tags	Artist terms and tags obtained from the Echo Nest API and the Music Brainz.	Using the terminal: rsync -avzuP publicdata.opensciencedatacloud.org::ark:/31807/osdc-c1c763e4/remotefile /path/to/local_copy **Located in file A	SQLite	1
Million Song Duplicates	List of known song duplicates.	http://labrosa.ee.columbia.edu/millionsong/sites/default/files/AdditionalFiles/msd_duplicates.txt	.txt	1
Million Song Mismatches	List of known song mismatches.	http://labrosa.ee.columbia.edu/millionsong/sites/default/files/tasteprofile/sid_mismatches.txt	.txt	1

DATA MANAGEMENT



DATA CLEANING: *SONG METADATA*

Step 1: Extract analysis metadata, song metadata and music brainz data from .h5 files using h5py.

Step 2: Convert data to pandas dataframes.

Step 3: Remove b‘ ’ strings.

Step 4: Replace zeroes with np.nan’s

DATA CLEANING: *ARTIST TERMS AND TAGS*

Step 1: The artist terms and tags are contained in a SQLite database.

Step 2: The database has five tables. The tables artist_mbtag and and artist_term were queried using sqlalchemy and converted to a pandas dataframe.

Step 3: The resulting dataframes were merged on artist id and using a left merge approach.

Step 4: Missing values in the merged dataframe were filled using the pandas method fillna.

DATA CLEANING: *USER LISTENING DATA*

Step 1: The user listening data text file was converted from text to a dataframe.

Step 2: The dataframe read the several columns of data as one. Separating by delimiter was used to separate data into several columns. The result was a dataframe with user_id, song_id and play_count.

Step 3: A column called "play" with value 1 was added to indicate the user played the song. The resulting dataframe has ~48.4M entries.

DATA CLEANING: *CONCAT USER, SONG METADATA & ARTIST TERMS*

Step 1: Remove known duplicates and mis-matches from the song metadata using LabRosa's lists.

Step 2: Merge all song metadata: analysis metadata, song metadata and music brainz data.

Step 3: Redundant columns created by the merge process were dropped. Blank entries were replaced with 'NaN' using two approaches: fillna and replace. The latter created two blank columns: analyzer version and genre. These blank columns were removed.

Step 4: Convert song metadata, artist tags and user listens dataframes to tidy format.

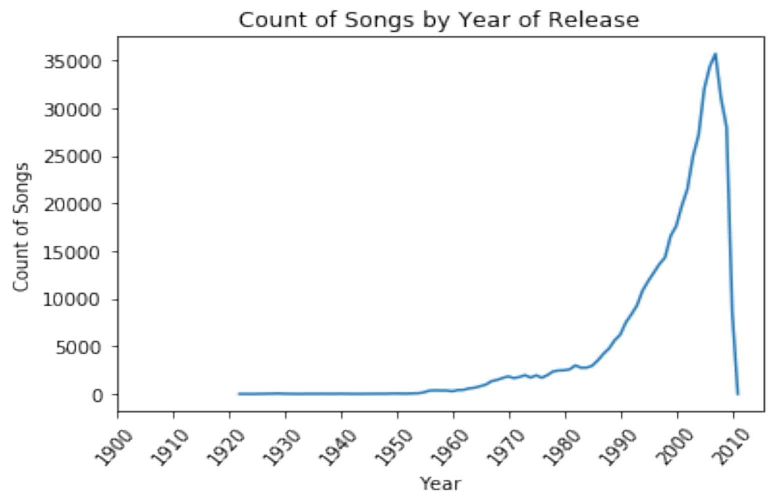
Step 5: Output song metadata, user listening data and artist tags as CSV files.

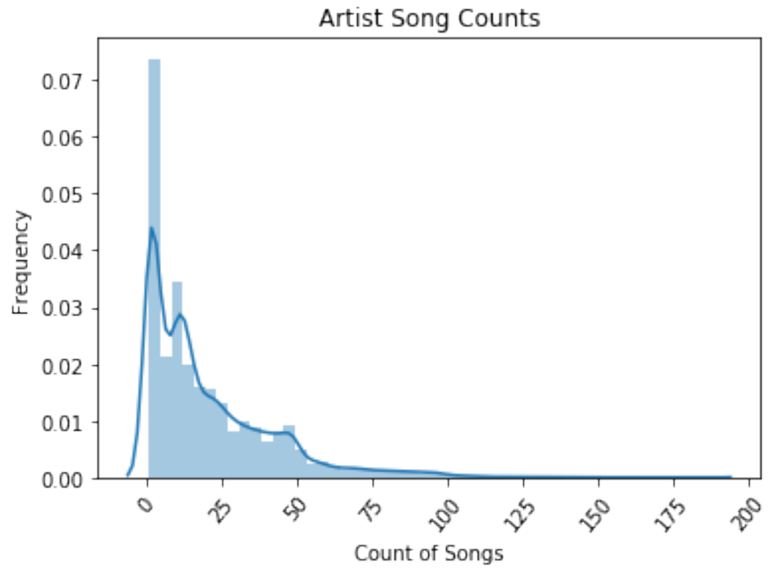
DATASETS FEATURES

Dataset	Feature Name	Description
User Listen Data	song_id	Echo Nest song ID
	user_id	Echo Nest user ID
	play_count	Number of times user played a song
	play	Whether a user played the song at least once. In recommender systems, it is best practice to treat implicit feedback as 0 or 1 (1 = consumed, 0 = not consumed)
Song Metadata	artist_7digitalid	ID from 7digital.com or -1
	artist_familiarity	algorithmic estimation
	artist_hotttnesss	algorithmic estimation
	artist_id	Echo Nest ID artist ID
	artist_latitude	latitude
	artist_location	longitude
	artist_mbid	ID from musicbrainz.org
	artist_name	artist name
	artist_playmeid	ID from playme.com, or -1
	release	album name
	release_7digitalid	ID from 7digital.com or -1
	song_hotttnesss	algorithmic estimation
	song_id	Echo Nest song ID
	title	song title
	track_7digitalid	ID from 7digital.com or -1
	audio_md5	audio hash code
	danceability	algorithmic estimation
	duration	in seconds
	energy	energy from listener point of view
	key	key the song is in
	loudness	overall loudness in dB
	mode	major or minor
	tempo	estimated tempo in BPM
	time_signature	estimate of number of beats per bar, e.g. 4
	track_id	Echo Nest track ID
	year	song release year from MusicBrainz or 0
Artists Tags	artist_id	Echo Nest artist id
	mbtag	Artists tags from musicbrainz.com
	terms	Artists terms from Echo Nest

EDA: *SONG METADATA*

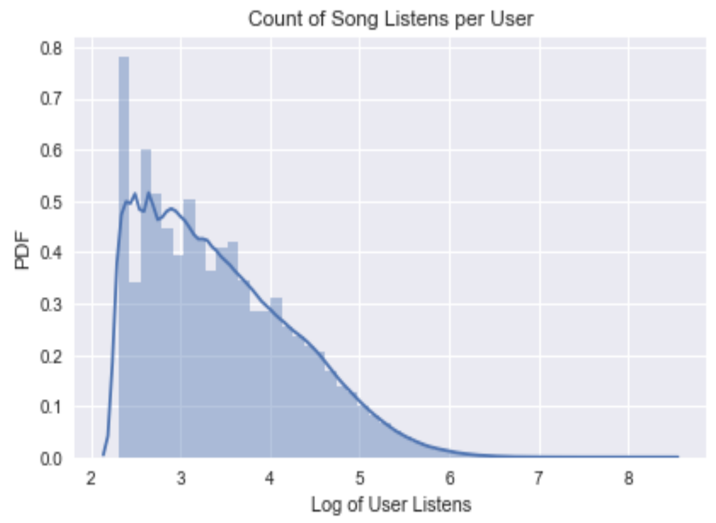
- 905,531 unique songs
- Peak year release of songs is 2000's.
- Artist counts distribution is right skewed with 3 peaks
- *Visuals: histograms, ECDF, bootstrap replicates*

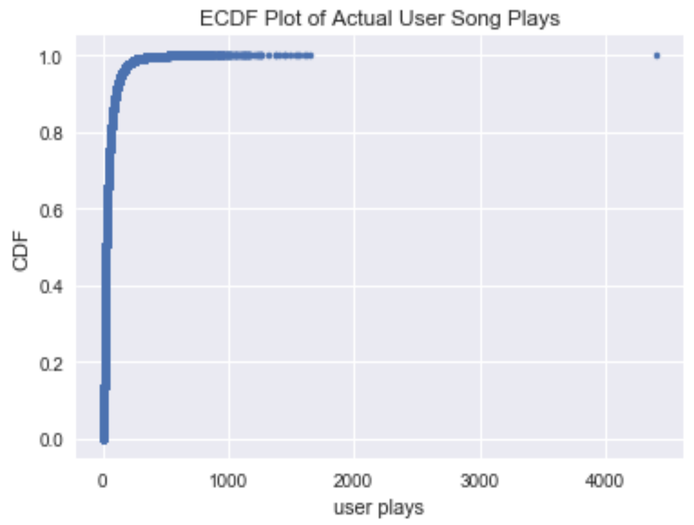




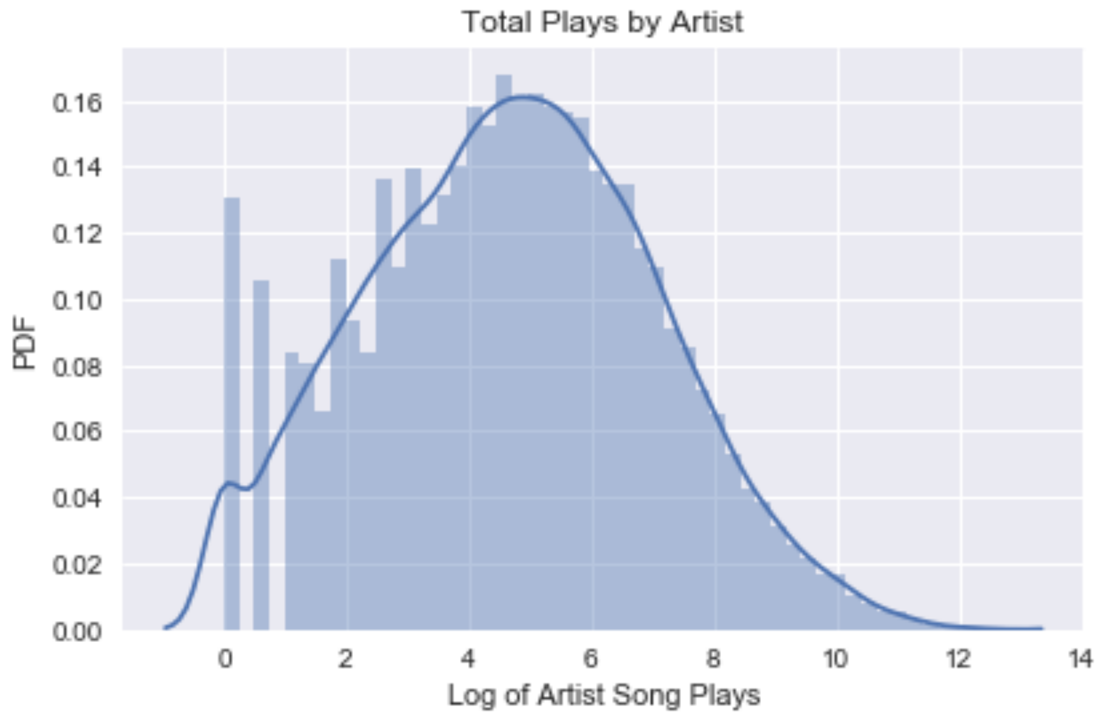
EDA: *USER LISTENING HISTORY*

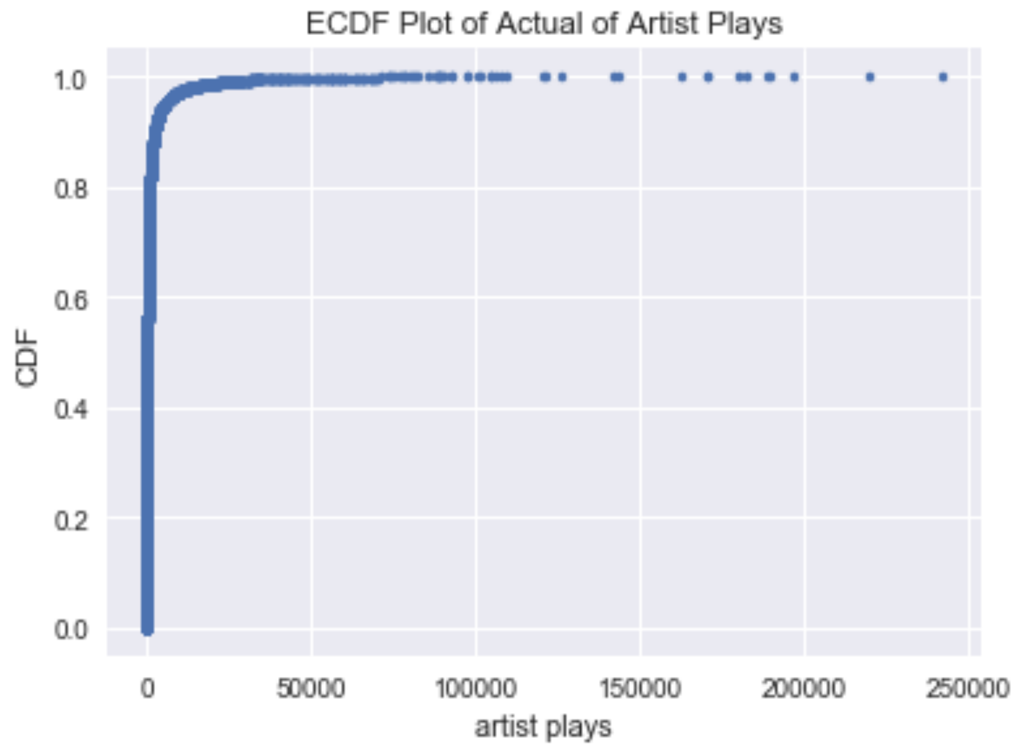
- 1.019M unique users
- 42% of song the catalog has been listened to
- 66.5% of artists available listened
- Distribution heavily skewed to the right
  - Skewness = 4.826
  - Kurtosis = 68.503
- *Visuals: histograms, ECDF, bootstrap replicates*





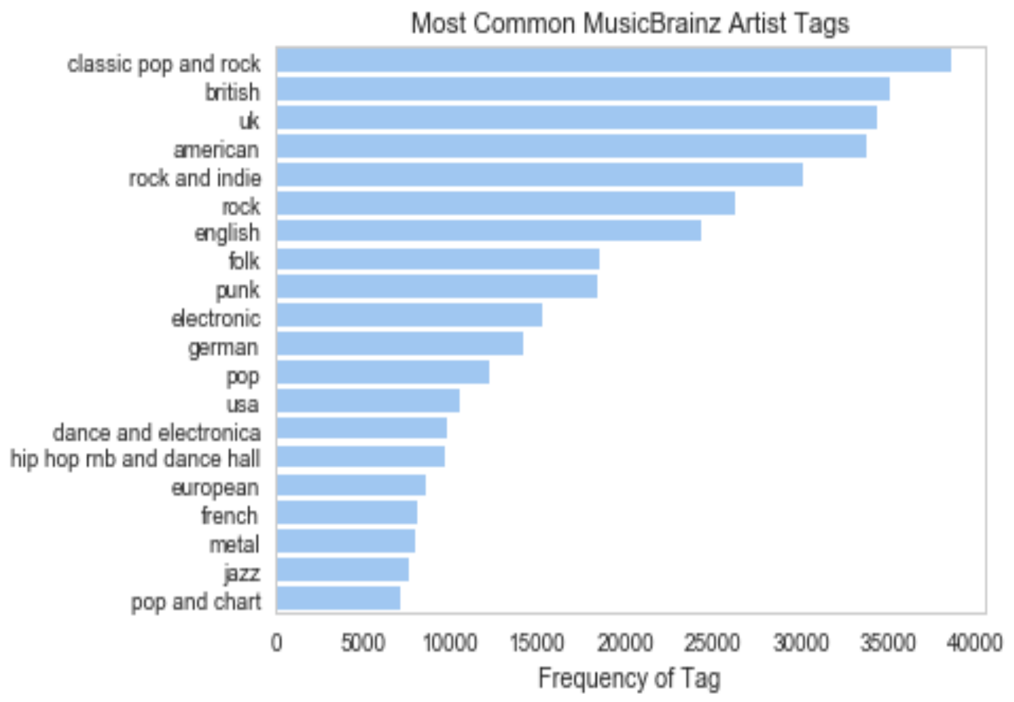
EDA: USER LISTENING HISTORY

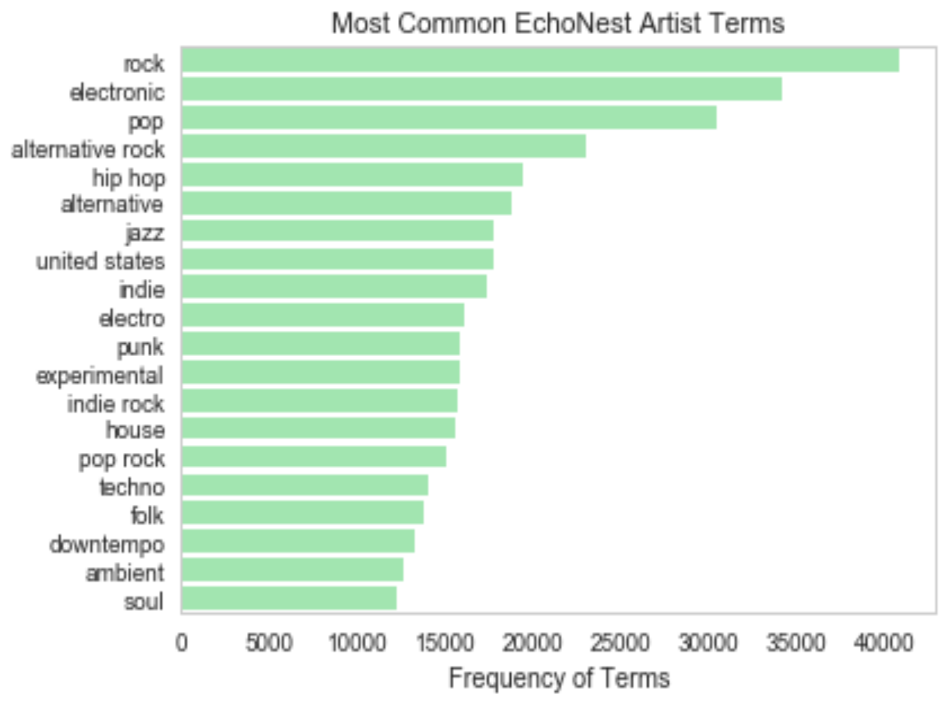




EDA: *ARTIST TAGS*

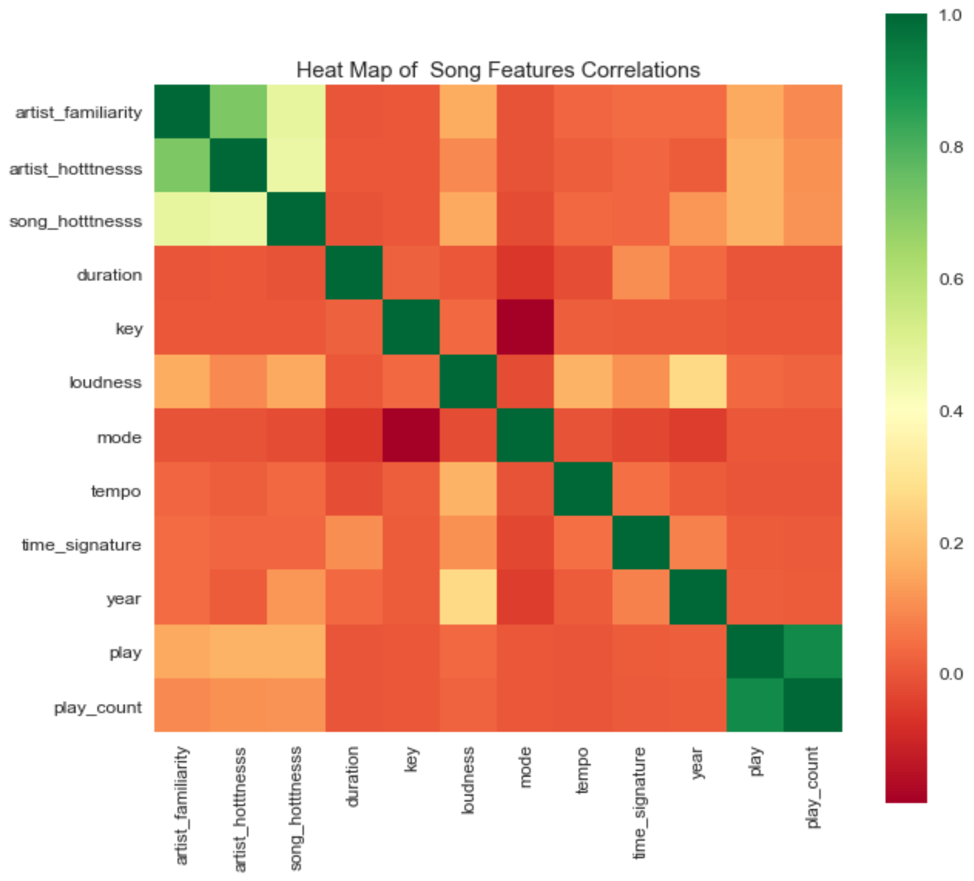
- Top tags and terms: rock, electronic and pop.





CORRELATIONS

No strong relationship found between listen history and song features.



FEATURE SELECTION & DATA PREPARATION

1. **User Data:** Filtered out users who: (1) have less than 45 unique song listens and, (2) users who have more than 452 song unique song listens.
   - *65.6% of the available user listening data was preserved for training and testing.*
2. **Song Data:** To reduce the memory costs, I only maintained columns that directly describe key aspects of the songs.
   - *artist_familiarity, artist_hotttnesss, duration, end_of_fade_in, key, loudness, start_of_fade_out, tempo, time_signature*
3. **Conversion to SFrames:** pandas Dataframes data were converted to GraphLab's SFrame format. The GraphLab's recommendation library only works with data in SFrames and SArrays formats.

FEATURE SELECTION & DATA PREPARATION

4. **Normalization:** Numerical features (0 – 1)
5. **Conversion to SFrames:** pandas Dataframes data were converted to GraphLab's SFrame format. The GraphLab's recommendation library only works with data in SFrames and SArrays formats.
6. **Random Shuffle:** Both the user and song data rows were randomly shuffled.
7. **Train-Test Splits:** Two train-test split samples were created: one of 80% train and 20% test, and a second one of 90% train and 10% test.
8. **Dataset for Artist Popularity Recommender:** A user dataset that contains the song's artist was created.

EVALUATION STRATEGY

- **Metric: Recall**

  ✓ *Main concern is returning relevant results to the user. Moreover, given that we are working with implicit data and not ratings or label classification precision can take a somewhat ambiguous nature.*

- **Recall (R)** is defined as the number of true positives (TP) over the number of true positives plus the number of false negatives (FN). Fraction of relevant instances that have been retrieved over the total number of relevant instances.

$$Recall = \frac{True\ Positives}{True\ Positives + Negatives}$$

MODELING: *FOUR RECOMMENDERS*

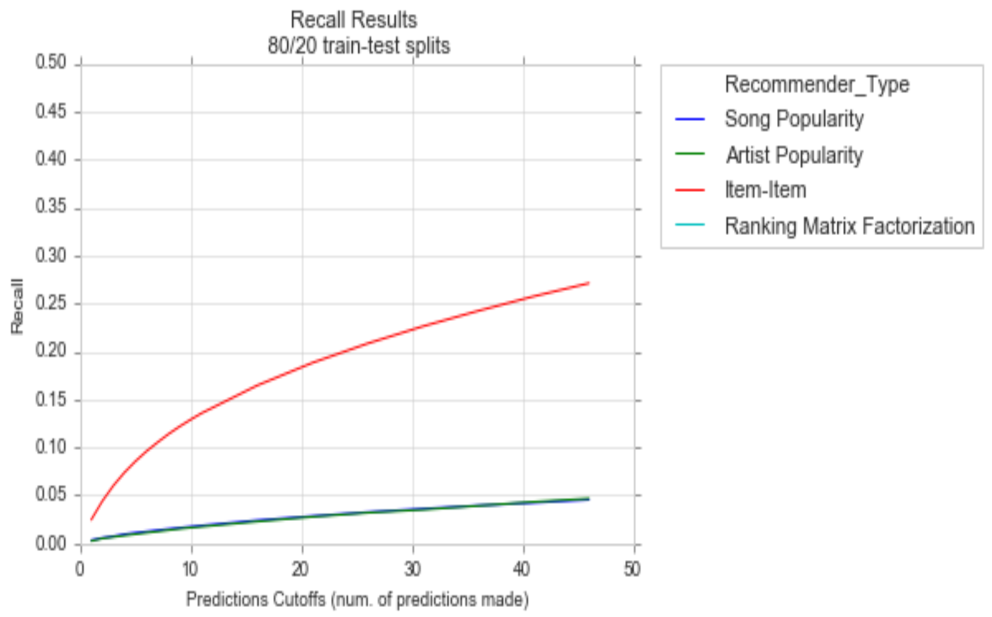
1. **Song Popularity:** The song popularity recommender scores songs based on the number of times they are seen in the dataset. The recommender returns the songs that are the most popular to all users and no personalization occurs. This type of recommender can be either used as a baseline or "background" model for new users.
2. **Artist Popularity:** The artist popularity recommender scores artists based on the number of times they were seen in the dataset. The recommender returns the artists that are the most popular to all users and no personalization occurs. This type of recommender can be used as a baseline or "background" model for new users.

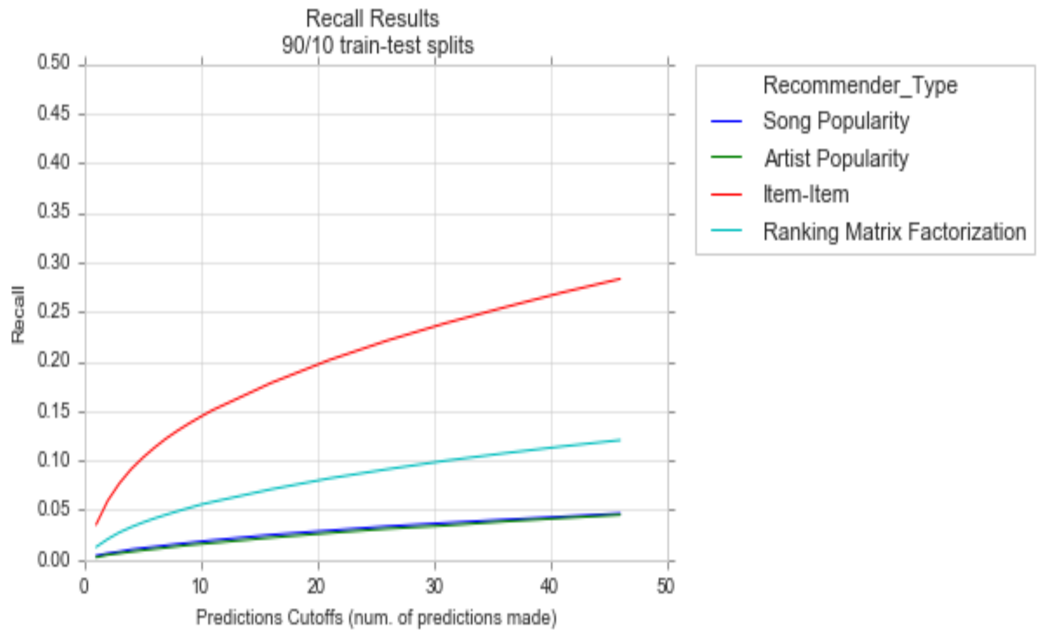
MODELING: *FOUR RECOMMENDERS*

3. **Item – Collaborative Filtering:** The item-based collaborative filtering recommender makes recommendations to users by identifying similar songs that were listened by similar users. It recommends similar songs that users have listened to, but that the target user has not necessarily listened to.
   - ✓ Similarity type: Jaccard
   - ✓ Number of Similar Items: 64
   - ✓ Nearest neighbor's interaction proportion: 0.001

4. **Ranking Matrix Factorization:** learns latent factors for each user by decomposing the user-song interactions. The decomposed matrix is them used to rank recommended items according to the probability of observing the same user, item pairs.
   - ✓ Regularization = 1e-9
   - ✓ Linear Regularization = 1e-9
   - ✓ Number of Latent Factors = 32
   - ✓ Iterations = 25
   - ✓ Solver: Implicit Alternating Least Squares

MODEL PERFORMANCE RESULTS





PERFORMANCE RESULTS SUMMARY

- ✓None of the recommenders recall performed better than 0.50. This means, on the whole they are not performing better than random.
- ✓ Item-based collaborative filtering recall performed significantly better than matrix factorization, song popularity and artist popularity recommenders.
- ✓ Both of the personalized recommenders performed better than popularity based recommenders.
- ✓ Recall performance was similar for the 80-20 and 90-10 train test samples.
- ✓ Recall performance increases with number of predictions made.

LIMITATIONS

1. **Computational Resources**

   - ***Content-Based Recommender:*** This was attempted and training would have been significant on this dataset even when not all song features were included. Given the fact that we have a lot of data on the songs and much less on the users, this may be a dataset that could benefit from a content-based strategy.
   - ***Five-Fold Cross-Validation:*** Computational resources were not sufficient to implementing the five different times. On the other hand, given the evaluation results there is no reason to be concerned about overfitting at this point.
   - ***Hyper-parameter tuning:*** The matrix factorization recommender and to a lesser extent has several items collaborative filtering has several parameters that can be set or modified to improve performance. In particular, the matrix factorization models are particularly sensitive to the regularization terms.

LIMITATIONS

2. **Evaluation Limited Usefulness**

   ✓Given that the data is implicit, a user listening to a song does not mean they necessarily liked it or that is their preferred genre. Moreover, we have no information on whether the 'listening' is skip or not.

   ✓Music listening can be idiosyncratic. Users listening can be driven by mood or context.

3. **Other Options**

   ✓Not all options for recommendation that may work for the business problem at hand was explored and tested.

RECOMMENDATIONS & NEXT STEPS

1. Explore a content-based strategy. This may be beneficial given the large amounts of information known about the songs.
2. The Ranking Matrix Factorization algorithm may have room for improvement in particular using parameter tuning, specially regularization terms and number of latent factors.
3. Include song metadata information in the matrix factorization algorithm.
4. Instead of ranking matrix factorization, it is possible to use a matrix factorization algorithm that treats the interactions as binary (0,1), and then implements a logistic regression to predict their binary rating.
5. Use a mix of methods. In a recommendation context, generally a mix of methods tends to outperform a single recommender. This is because we can exploit the strengths of each recommender and exploit different aspects of the data with each recommender.

RECOMMENDATIONS & NEXT STEPS

6. Clustering is another option that maybe worth exploring. However, given the sparsity and heavy skewness of the data, it is a question of whether meaningful and predictive user clusters could be achieved.
7. Do more investigation of user listening behavior. It is not necessarily true that if someone has ‘listened’ to a song they necessarily like it. Also, if there if many of the listening that are actually ‘skips’ that could have a large impact on the performance of the recommender.
8. Benchmark performance. In this implementation, I have used recall as the evaluation metric, however due to the limitations of the data and idiosyncratic nature music listening could take, it may be fruitful to benchmark with other music recommendation systems performance to gain a sense of what is reasonable in this context.

SUMMARY

- Data on song metadata, music listening history for 1M users and artists tags was obtained from LabRosa at Columbia University
- Significant data wrangling was required to convert the raw data to an appropriate format for analysis.
- Four recommenders were tested: song popularity, artist popularuty, item-item, and matrix factorization.
- Computational resources were limited as such, hyperparameter tuning, cross-validation and a content-based recommender were not possible.
- Significant room for improvement remains, none of the recommenders is performing better than random in terms of recall.