E 4571 - Personalization Theory Project; Part2; Fall-2018

Team: Phani Valasa & Harish Visweswaran

Data Science Institute, Columbia University in the City of New York

Recommender System Project:

The main purpose of this project is to design a recommendation engine that has the ability to power a streaming music service by providing useful song recommendations for its users.

Table of Contents

Business Abstract and Motivation:

The objective of this Case study is to improvise online music provider services. Many of the Music service providers today offer a freemium model with the majority of its users streaming music to their mobiles or desktop via apps or web browsers. Users subscribe for free get ads between tracks which are part of the revenue model. Users of the free service encounter audio ads every five or six songs, or approximately three minutes of advertising for every hour of listening.

There are also paid services with uninterrupted streaming. For this business plan to work and remain relevant, it is imperative that right recommendations on songs are provided to users for customer loyalty and stickiness with the app. The problem is more complex than it seems. Challenges are with the sparse dataset pertaining to user listening and identification of similarities between users, songs and artists.

The below case study accesses the data sets available from Kaggle, MSD, LabROSA and Last.fm to evaluate the recommendation models and build a recommendation engine for our Music streaming company.

Objectives of this project:

The primary objective of this work is to create a contextually relevant and meaningful recommender system to recommend songs to the users for an online/mobile music streaming platform. Our goal is also to explore and gain practical experience with modern personalization frameworks, especially the following approaches.

1. Approximate Nearest Neighbors: Implement and evaluate a native Locality Sensitive Hashing (LSH) Model to recommend songs.

2. Hybrid Matrix Factorization: Obtain effective recommendations using hybrid matrix factorization method, implemented by the LightFM library.

The above approaches are explained as separate sections below.

Approximate Nearest Neighbors:

Overview of Locality Sensitive Hashing:

As we've seen earlier, the challenges in memory based collaborating filtering is the time and memory required because of High dimensionality of the feature vector (user listens) of songs and the comparison against features of all songs in the inventory.

The first challenge can be addressed using a dimensionality reduction technique like PCA and the second using a combination of clustering and nearest neighbor search. Locality Sensitive Hashing (hereon referred to as LSH) can address both.

Locality-sensitive hashing (LSH) reduces the dimensionality of high-dimensional data. LSH hashes input items so that similar items map to the same "buckets" with high probability (the number of buckets being much smaller than the universe of possible input items). LSH differs from conventional and cryptographic hash functions because it aims to maximize the probability of a "collision" for similar items. Locality-sensitive hashing has much in common with data clustering and nearest neighbor search.

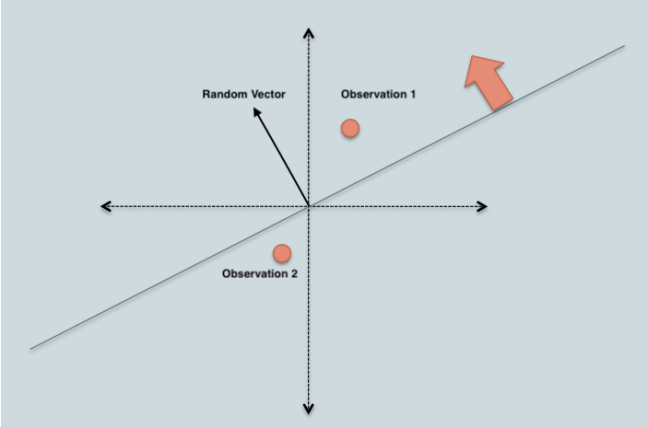
The LSH algorithm has been applied successfully in large data sets to find nearest neighbors quickly. Instead of find exact matches as conventional hashes would, LSH considers locality of points so that nearby points remain nearby.

Our Implementation of LSH:

Random Projection:

Random projection is a technique for representing high-dimensional data in low-dimensional feature space. The implementation is based on a simple idea that if two points are close to each other, then after a projection these points remain close together.

At the core of LSH is the scalar projection (or dot product), given by $h(v) = v \cdot x$, where v is a query point in a high-dimensional space, and x is a vector with components that are selected at random from a Gaussian distribution, for example N(0, 1). Below is an illustration on how we are considering our projections to obtain a binary value.

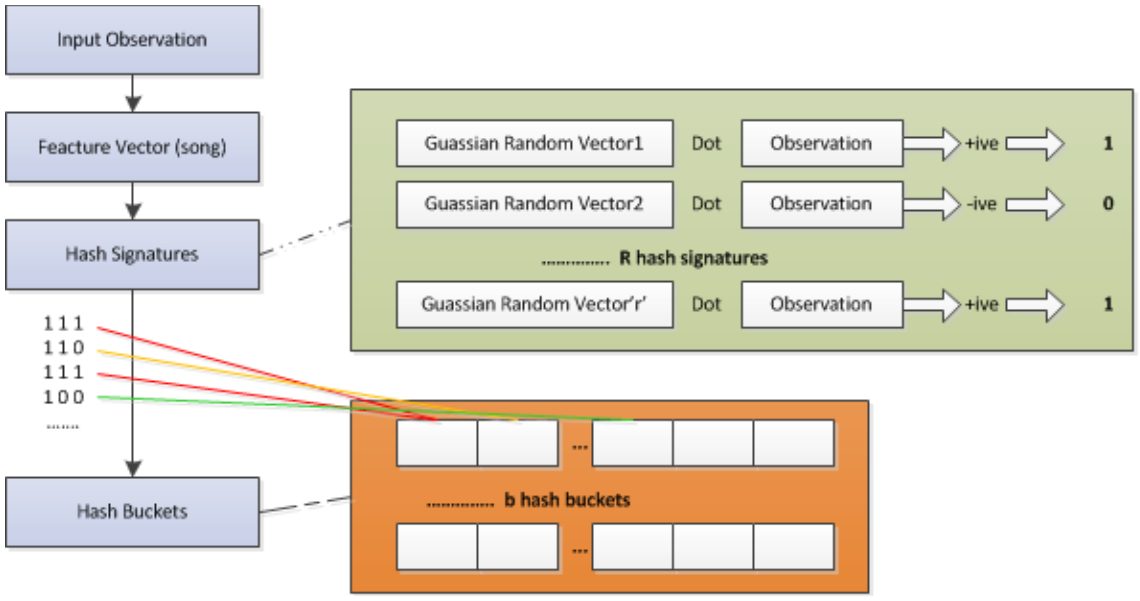


The dot product of Observation 1 and Random vector gives a value > 0, so we consider the output signature as 1. Similarly for Observation 2, we consider the output signature 0 since the dot product value is < 0.

This scalar projection is then quantized into a set of hash bins, with the intention that nearby items in the original space will fall into the same bin. We can further magnify the differences between P1 and P2, by performing r dot products in parallel.

Algorithm used to identify similar items:

- Create 'r' random Gaussian vector each having length equal to the dimension of feature vector (i.e., for every song)
- For each random vector, compute the dot product with Observation. If the result of the dot product is positive, assign 1 as signature. Else the signature value is 0
- Concatenative all the signatures obtained for r dot products. Calculate the hash value by converting the binary signature to decimal value.
- Repeat the above two steps for all Observations and compute hash values.
- Group the Observations with same has values in a LSH hash table bucket.
- Repeat the above for 'b' buckets.



LSH Time Complexity:

Time Complexity to find out most similar items for a given item is O(n). To create an offline model for similar item pairs of all items is O(n2). This is same as Item based collaborative filtering which will be significantly larger when we have huge data sets. In order to overcome this challenge, approximate nearest neighbor techniques (LSH in this case) generates small subsets of items which have higher likelihood of being similar to the items under consideration.

Datasets:

We've run our models on two sample datasets. In order to show the improvements/detriment in time taken to build models, we created another sample of dataset with 5000 songs and 10,000 users. Also shown are the details on sparsity.

DataSet	#songs	#users	#user-song pairs	Density (1-Sparsity)
500 Songs - Entire Dataset	500	9,943	357,505	7%
500 Songs - Train Dataset (80%)	500	9,912	286,004	6%
500 Songs - Test Dataset (20%)	500	9,221	71,501	NA
5000 Songs - Entire Dataset	5,000	10,000	1,394,964	3%
5000 Songs - Train Dataset (80%)	5,000	9,999	1,115,971	2%
5000 Songs - Test Dataset (20%)	5,000	9,985	278,993	NA

Please note that in the metrics evaluation section below, we present them separately for each sampled dataset we have used. This is because we wanted to explicitly show the difference in compute time and model performance as the data size is increased.

Baseline Metrics:

Below are the baseline metrics on our two data sets. Baseline prediction of listen count is calculated as the average values from the listen counts on Train data set. And this is used to calculate the RMSE, MAE and MSE on Test data set. We can see how these metrics compare with all the LSH and Item CF models that we created. In all cases, the performance on our LSH and Item CF models is better than this baseline.

Baseline Metrics

DataSet	RMSE	MAE	MSE
500 Songs - Entire Dataset	4.30372	2.00613	18.52199
5000 Songs - Entire Dataset	3.42790	1.63690	11.75047

Metrics Evaluation: Dataset with 10,000 users and 500 songs:

Model Comparison Metrics:

We've run a total of 63 models (LSH and Item Based Collaborative filtering) on our 500 songs - 10,000 user dataset to compare the performance metrics and coverage. Here are the details on our top 30 models ordered by RMSE. The hyper parameters - R is the number of hash functions, B is the number of hash buckets and K is the neighborhood set size for prediction. Also included are the RMSE, MAE, MSE and Coverage for various top N predictions on the Test/Hold-out dataset. As it can be seen, the best metrics (RMSE, MAE, MSE) are on Item Based Collaborative filtering while the LSH models are performing close enough with much better improvement in compute time.

Coverage is a measure of the domain of items over which recommendations can be made by the system. Coverage metrics that we presented is the percentage of (user, song) pairs from our Test dataset that are recommended by that Model. As we increase the number of predictions the likelihood of the items on the test data increase, which can be noticed in our results too.

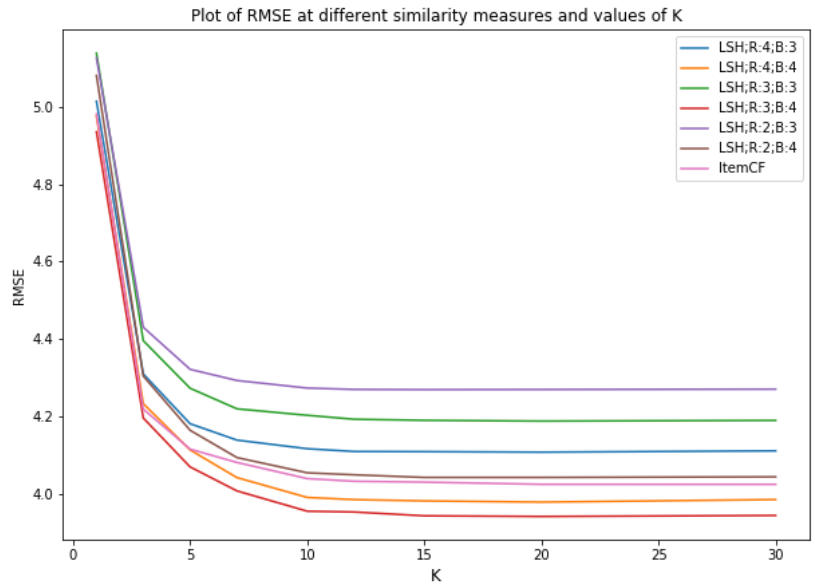
Model	R	B	K	Time (Secs)	RMSE	MAE	MSE	Top20 Coverage	Top40 Coverage	Top60 Coverage
ItemCF			20	51.44	3.942739	1.748806	15.54519	12.68%	20.12%	25.94%
ItemCF			15	51.44	3.944426	1.754876	15.5585	12.60%	19.94%	25.74%
ItemCF			30	51.44	3.945312	1.741979	15.56549	12.78%	20.31%	26.14%
ItemCF			12	51.44	3.954271	1.759164	15.63626	12.34%	19.72%	25.39%
ItemCF			10	51.44	3.956287	1.761789	15.65221	12.12%	19.39%	25.13%
LSH	2	4	20	38.64	3.979805	1.752073	15.83885	10.84%	17.76%	23.52%
LSH	2	4	15	38.64	3.982908	1.758644	15.86355	10.89%	17.82%	23.42%
LSH	2	4	12	38.64	3.986157	1.763263	15.88945	10.83%	17.79%	23.31%
LSH	2	4	30	38.64	3.986309	1.748973	15.89066	10.66%	17.79%	23.55%
LSH	2	4	10	38.64	3.991809	1.766233	15.93454	10.77%	17.59%	23.23%
ItemCF			7	51.44	4.008654	1.769851	16.06931	11.67%	18.78%	24.61%
LSH	2	3	30	33.27	4.025566	1.762932	16.20518	9.53%	16.29%	21.90%
LSH	2	3	20	33.27	4.025707	1.770005	16.20632	9.80%	16.41%	22.22%
LSH	2	3	15	33.27	4.031301	1.777086	16.25139	9.96%	16.68%	22.30%
LSH	2	3	12	33.27	4.03353	1.782607	16.26936	9.89%	16.73%	22.34%

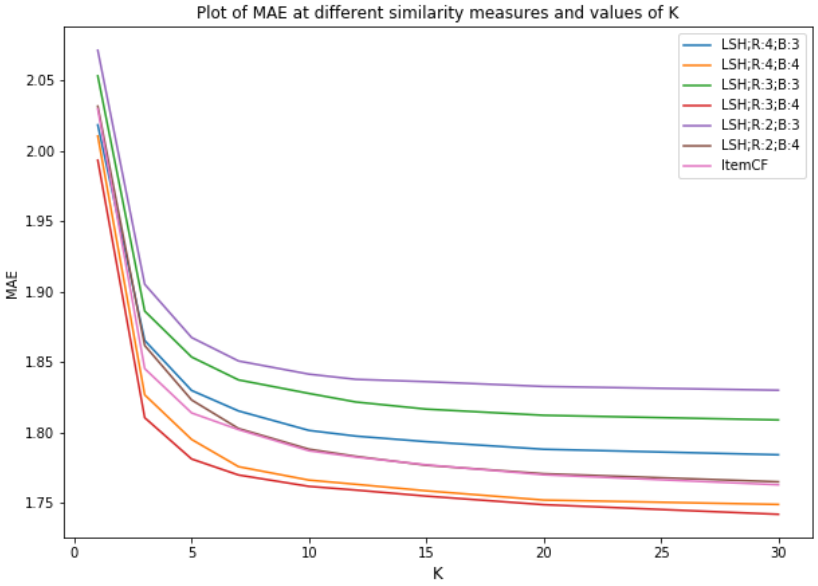
LSH	2	3	10	33.27	4.040386	1.787052	16.32472	9.91%	16.63%	22.36%
LSH	2	4	7	38.64	4.042911	1.77576	16.34513	10.53%	17.34%	22.85%
LSH	3	4	20	27.45	4.043453	1.770786	16.34951	8.75%	15.30%	20.85%
LSH	3	4	15	27.45	4.043486	1.776746	16.34978	9.00%	15.44%	21.16%
LSH	3	4	30	27.45	4.044975	1.765044	16.36182	8.42%	14.91%	20.47%
LSH	3	4	12	27.45	4.050171	1.78311	16.40388	9.05%	15.51%	21.08%
LSH	3	4	10	27.45	4.055362	1.78819	16.44596	9.13%	15.54%	21.15%
ItemCF			5	51.44	4.070501	1.781222	16.56898	11.16%	17.98%	23.70%
LSH	2	3	7	33.27	4.082007	1.802045	16.66278	9.73%	16.48%	22.14%
LSH	3	4	7	27.45	4.094621	1.80285	16.76592	9.13%	15.44%	21.09%
LSH	3	3	20	22.28	4.10862	1.788127	16.88076	7.37%	13.48%	18.88%
LSH	3	3	15	22.28	4.109948	1.793514	16.89167	7.62%	13.75%	19.18%
LSH	3	3	12	22.28	4.110357	1.79745	16.89503	7.80%	13.98%	19.41%
LSH	3	3	30	22.28	4.111965	1.784215	16.90825	7.12%	13.14%	18.55%
LSH	2	4	5	38.64	4.11428	1.795078	16.9273	10.10%	16.81%	22.31%

Here are the different values of K-neighbors on LSH and ItemCF models for which the RMSE is the least. It is to be noted that both LSH and ItemCF models have got the best RMSE for the value of K at 20. We present the compute time on various models, using this value of K=20, later in this section.

Model	K	min(RMSE)	min(MAE)	min(MSE)	Top10 Coverage	Top20 Coverage	Top40 Coverage	Top60 Coverage
ItemCF	10	3.956287	1.761789	15.65221	7.40%	12.12%	19.39%	25.13%
LSH	10	3.991809	1.766233	15.93454	6.44%	10.77%	17.59%	23.23%
ItemCF	12	3.954271	1.759164	15.63626	7.56%	12.34%	19.72%	25.39%
LSH	12	3.986157	1.763263	15.88945	6.48%	10.83%	17.79%	23.31%
ItemCF	15	3.944426	1.754876	15.5585	7.76%	12.60%	19.94%	25.74%
LSH	15	3.982908	1.758644	15.86355	6.48%	10.89%	17.82%	23.42%
ItemCF	20	3.942739	1.748806	15.54519	7.82%	12.68%	20.12%	25.94%
LSH	20	3.979805	1.752073	15.83885	6.46%	10.84%	17.76%	23.52%
ItemCF	30	3.945312	1.741979	15.56549	7.79%	12.78%	20.31%	26.14%
LSH	30	3.986309	1.748973	15.89066	6.34%	10.66%	17.79%	23.55%

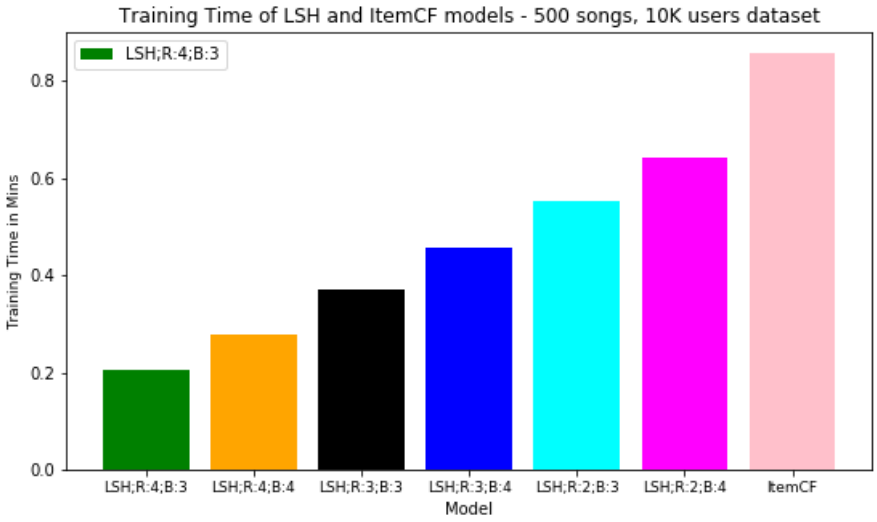
The below figures shows RMSE and MAE on the test dataset using several LSH and ItemCF models. They depict how RMSE and MAE are changing for different values of K.





Time Complexity:

Here is a comparison on time taken to build the similarity matrix using various LSH models and Item based Collaborative filtering. As it can be seen that some of LSH models are 5 times faster than Item Based CF. This was run on a 500 songs and 10,000 users datasets (reduced the size due to the computation limitation on the laptop). While the item based CF is taking about a minute to compute similarity matrix, best time on LSH models is close to 15 seconds. As the number of items increases in the dataset, LSH models could be much more effective. We'll show the results from a 5,000 songs and 10,000 user dataset as well later in this report.



Predictions- Relevance, Novelty, Serendipity and comparison with Item based CF model:

Novelty is a measure of serendipity. A generally subjective understanding expresses an item as novel if it is much different from the items usually seen. Another definition of novelty is defined in terms of the amount of information its observation conveys and is defined as the negative log of its corresponding probability. In this report, we evaluate novelty by judging recommendations on a qualitative basis. In this section, we present the recommendations for a user using LSH and Item Based CF to compare them and assess Novelty and relevance.

Below are the predictions for a given user (User ID - 6a944bfe30ae8d6b873139e8305ae131f1607d5f) using LSH and Item based Collaborative filtering (the models with best RMSE). When we look at LSH Top 20 predictions with no filter on already listened songs, 60% of the predictions include songs that have been already listened to which indicates our predictions are relevant and not totally off. This is the observed pattern on both LSH and Item based CF models.

Also notice that there are a lot of common recommendations between LSH and Item CF models in our Top20 output. There is a 90% overlap when the listened songs were not filtered and 85% overlap when listened songs are excluded. So the models are comparable not just in terms of RMSE, MAE and MSE performance but also in terms of qualitative checks on the songs recommended. We are get similar predictions using LSH while reducing the training compute time significantly.

It is also noted that there are some predictions belonging to a different genre altogether, so our predictions does have a bit of novelty factor in it.

LSH - Top 20 Predictions including Listened Songs			
Listen Count prediction	artist_name	track_name	Listened
32	Sam Cooke	Ain't Misbehavin	Y
23	Björk	Undo	Y
22	Dwight Yoakam	You're The One	Y
17	Coldplay	Lost!	Y
15	Godsmack	Speak	N
15	Irene Cara	What A Feeling	N
15	Hot Chip	Bad Luck	N
14	Tub Ring	Invalid	Y
14	Sheena Easton	Strut (1993 Digital Remaster)	N
13	Future Rock	Gears	N
13	Pavement	Mercy:The Laundromat	N
13	Barry Tuckwell/Academy o	Horn Concerto No. 4 in E flat K495:	Y
13	Usher featuring will.i.am	OMG	Y
13	OneRepublic	All The Right Moves	N
13	Train	Marry Me	Y
13	Kid Cudi / MGMT / Ratatat	Pursuit Of Happiness (nightmare)	Y
13	Alliance Ethnik	Représente	Y
13	Kings Of Leon	Revelry	N
13	Lonnie Gordon	Catch You Baby (Steve Pitron & Ma	Y
12	Train	Hey_ Soul Sister	Y

Common Predictions: 18/20 (90% Overlap)

Item CF - Top 20 Predictions including Listened Songs			
Listen Count prediction	artist_name	track_name	Listened
32	Sam Cooke	Ain't Misbehavin	Y
23	Björk	Undo	Y
23	Dwight Yoakam	You're The One	Y
17	Coldplay	Lost!	Y
15	Godsmack	Speak	N
15	Irene Cara	What A Feeling	N
14	Tub Ring	Invalid	Y
14	Sheena Easton	Strut (1993 Digital Remaster)	N
13	Pavement	Mercy:The Laundromat	N
13	Barry Tuckwell/Academy of	Horn Concerto No. 4 in E flat K49	Y
13	OneRepublic	All The Right Moves	N
13	Usher featuring will.i.am	OMG	Y
13	Train	Marry Me	Y
13	Kid Cudi / MGMT / Ratatat	Pursuit Of Happiness (nightmare	Y
13	Alliance Ethnik	Représente	Y
13	Kings Of Leon	Revelry	N
13	Bon Jovi	Livin' On A Prayer	N
13	Future Rock	Gears	N
13	Lonnie Gordon	Catch You Baby (Steve Pitron & M	Y
13	Florence + The Machine	Dog Days Are Over (Radio Edit)	Y

LSH - Top20 Predictions Excluding Listened Songs			
Listen Count prediction	artist_name	track_name	Listened.
15	Godsmack	Speak	N
15	Irene Cara	What A Feeling	N
15	Hot Chip	Bad Luck	N
14	Sheena Easton	Strut (1993 Digital Remaster)	N
13	Future Rock	Gears	N
13	Pavement	Mercy:The Laundromat	N
13	OneRepublic	All The Right Moves	N
13	Kings Of Leon	Revelry	N
12	Beyoncé	Halo	N
12	Travie McCoy	Billionaire [feat. Bruno Mars] (Expl	N
12	Nickelback	How You Remind Me	N
12	Jason Derulo	Whatcha Say	N
12	N.E.R.D.	Rock Star	N
12	Bon Jovi	Livin' On A Prayer	N
12	Drake / Kanye West / Lil Wa	Forever	N
12	Randy Crawford	Almaz	N
12	Kid Cudi	Up Up & Away	N
12	J. Karjalainen & Mustat Lasi	Sinisten tähtien alla	N
12	Black Eyed Peas	Imma Be	N
12	California Swag District	Teach Me How To Dougie	N

Common Predictions: 17/20 (85% Overlap)

Item CF - Top20 Predictions Excluding Listened Songs			
Listen Count prediction	artist_name	track_name	Listened
15	Godsmack	Speak	N
15	Irene Cara	What A Feeling	N
14	Sheena Easton	Strut (1993 Digital Remaster)	N
13	Pavement	Mercy:The Laundromat	N
13	OneRepublic	All The Right Moves	N
13	Kings Of Leon	Revelry	N
13	Bon Jovi	Livin' On A Prayer	N
13	Future Rock	Gears	N
12	Beyoncé	Halo	N
12	Travie McCoy	Billionaire [feat. Bruno Mars] (Ex	N
12	Jason Derulo	Whatcha Say	N
12	N.E.R.D.	Rock Star	N
12	Nickelback	How You Remind Me	N
12	John Mayer	Heartbreak Warfare	N
12	Drake / Kanye West / Lil Way	Forever	N
12	Black Eyed Peas	Imma Be	N
12	Rihanna	Take A Bow	N
12	California Swag District	Teach Me How To Dougie	N
11	Kid Cudi	Up Up & Away	N
11	The Crests	16 Candles	N

Metrics Evaluation: Dataset with 10,000 users and 5,000 songs:

Model Comparison Metrics:

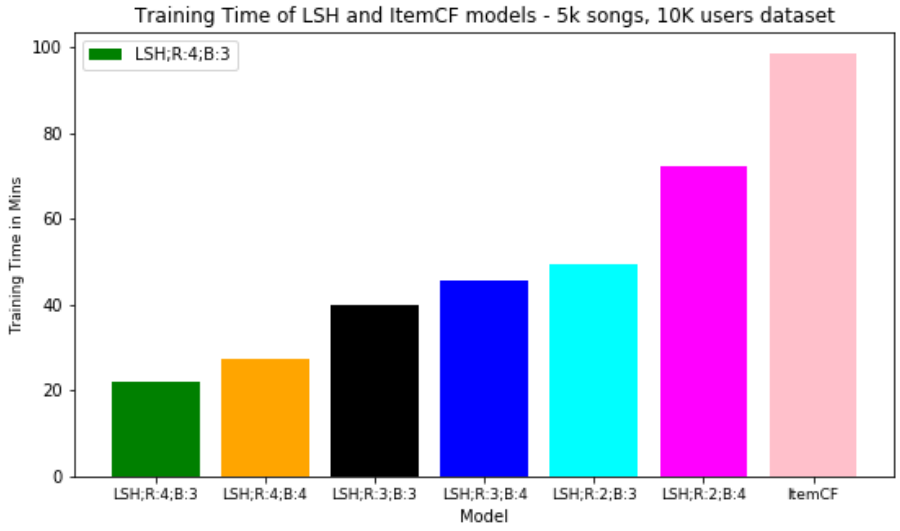
For a chosen K = 20 (neighborhood set), we have run the LSH and Item based CF on a training dataset with 1.12 million records which has 10,000 users and 5,000 songs. Below is a comparison of time taken to create similarity matrix and the corresponding model performance results. Coverage percentages are much lower in this case when compared to 500 songs dataset as we have very few recommendations made from a bigger dataset.

Model	R	B	K	Time (Mins)	RMSE	MAE	MSE	top 20 Coverage %	top 40 Coverage %	top 60 Coverage %
ItemCF			20	98.48	3.150446	1.423779	9.92531	2.29	3.77	5.00
LSH	2	4	20	72.24	3.165142	1.427074	10.01812	1.95	3.30	4.43
LSH	2	3	20	49.46	3.18901	1.434701	10.16978	1.85	3.16	4.26
LSH	3	4	20	45.73	3.200073	1.440491	10.24047	1.65	2.85	3.91

LSH	3	3	20	39.90	3.226849	1.447648	10.41256	1.42	2.52	3.49
LSH	4	4	20	27.33	3.279678	1.453818	10.75629	1.10	1.98	2.82
LSH	4	3	20	22.10	3.303475	1.470664	10.91294	0.96	1.77	2.56

Time Complexity:

As can be seen, the training time is 5 times faster using LSH in some models. This is similar to what we have seen in 500 songs dataset. However the difference in training time can be noticed. The Item based CF model took approx. 100 minutes while the LSH models took an average of 40 minutes.



Predictions- Relevance, Novelty, Serendipity and comparison with Item based CF model:

The quality of predictions in LSH model is comparable to item CF model as we've seen in our 500 songs dataset. Below are the predictions for a given user (User ID - 6a944bfe30ae8d6b873139e8305ae131f1607d5f) on our 5,000 songs dataset using LSH and Item based Collaborative filtering (the models with best RMSE). Here we have 30% of already listened songs in our Top 20 predictions for this user, showing that our predictions are relevant and meaningful. Similar pattern observed for both Item based CF and LSH models.

Also as we've seen in our 500 songs dataset results, there is an overlap of up to 85% in predictions between LSH and Item CF models.

LSH-Top 20 Predictions Including Listened Songs (5k Songs Dataset)

Listen Count prediction	artist_name	track_name	Listened
175	George Younce	This Old House w/ When The Saint	N
72	Daughtry featuring Slash	What I Want	N
54	Radiohead	Pop Is Dead	N
43	Harry Gregson-Williams	Bullet Tells The Truth	N
43	Amos Lee	Kid	N
42	Jack Johnson	People Watching	N
37	OneRepublic	Say (All I Need)	Y
37	Jack Johnson	Moonshine	N
37	Björk	Army of Me	N
36	Paramore	Ignorance (Album Version)	Y
35	Cutting Crew	(I Just) Died In Your Arms	Y
35	Colbie Caillat	I Never Told You	Y
33	D-12	Purple Pills	Y
32	Nick Cave & The Bad Seeds	Red Right Hand	N
30	Iration	Time Bomb	N
29	Jem	Falling for You	N
26	Jack Johnson	Holes To Heaven	N
26	OneRepublic	Good Life	N
25	Triple Six Mafia	Now I'm High_ Really High	N
24	Portishead	Machine Gun	N

Common Predictions: 16/20 (80% Overlap)

Item CF-Top 20 Predictions Including Listened Songs(5k Songs Dataset)

Listen Count prediction	artist_name	track_name	Listened
154	George Younce	This Old House w/ When The Saint	N
61	Daughtry featuring Slash	What I Want	N
52	Radiohead	Pop Is Dead	N
43	Reality Check	Masquerade (Reality Check Album	N
43	The Hives	Bigger Hole To Fill	N
38	Paramore	Ignorance (Album Version)	Y
37	OneRepublic	Say (All I Need)	Y
37	Cutting Crew	(I Just) Died In Your Arms	Y
37	Amos Lee	Kid	N
36	Colbie Caillat	I Never Told You	Y
36	Jack Johnson	Moonshine	N
35	Ryan Adams	Wonderwall	Y
35	D-12	Purple Pills	Y
34	Sam Cooke	Ain't Misbehavin	Y
33	Jack Johnson	People Watching	N
31	Harry Gregson-Williams	Bullet Tells The Truth	N
30	Björk	Army of Me	N
29	Nick Cave & The Bad Seeds	Red Right Hand	N
29	Jem	Falling for You	N
29	Iration	Time Bomb	N

LSH-Top 20 Predictions Excluding Listened Songs (5k Songs Dataset)

Listen Count prediction	artist_name	track_name	Listened.
175	George Younce	This Old House w/ When The Saint	N
72	Daughtry featuring Slash	What I Want	N
54	Radiohead	Pop Is Dead	N
43	Harry Gregson-Williams	Bullet Tells The Truth	N
43	Amos Lee	Kid	N
42	Jack Johnson	People Watching	N
37	Jack Johnson	Moonshine	N
37	Björk	Army of Me	N
32	Nick Cave & The Bad Seeds	Red Right Hand	N
30	Iration	Time Bomb	N
29	Jem	Falling for You	N
26	Jack Johnson	Holes To Heaven	N
26	OneRepublic	Good Life	N
25	Triple Six Mafia	Now I'm High_ Really High	N
24	Portishead	Machine Gun	N
24	Nine Inch Nails	Wish	N
23	Radiohead	Like Spinning Plates	N
22	Drowning Pool	Nothingness	N
21	Jack Johnson	Questions	N
21	Björk	Come To Me	N

Common Predictions: 17/20 (85% Overlap)

Item CF-Top 20 Predictions Excluding Listened Songs(5k Songs Dataset)

Listen Count prediction	artist_name	track_name	Listened
154	George Younce	This Old House w/ When The Saint	N
61	Daughtry featuring Slash	What I Want	N
52	Radiohead	Pop Is Dead	N
43	Reality Check	Masquerade (Reality Check Album	N
43	The Hives	Bigger Hole To Fill	N
37	Amos Lee	Kid	N
36	Jack Johnson	Moonshine	N
33	Jack Johnson	People Watching	N
31	Harry Gregson-Williams	Bullet Tells The Truth	N
30	Björk	Army of Me	N
29	Nick Cave & The Bad Seeds	Red Right Hand	N
29	Jem	Falling for You	N
29	Iration	Time Bomb	N
25	OneRepublic	Good Life	N
24	Drowning Pool	Cast Me Aside	N
24	Portishead	Machine Gun	N
24	The Black Keys	Have Mercy On Me	N
24	Triple Six Mafia	Now I'm High_ Really High	N
23	Jack Johnson	Holes To Heaven	N
23	Nine Inch Nails	Wish	N

Hybrid Matrix Factorization:

Overview and Algorithm Used:

For the same business problem of song recommendations, we wanted to use an algorithm that can be used for effective recommendations (without worrying about predicting the number of times the user will listen to a song). We came across the LightFM library which uses a hybrid matrix factorization model that represents users and items as linear combinations of their content features' latent factors.

Song consumption datasets are usually quite sparse – this is expected as there are tons of songs available to users but an average user will listen only to a small subset of songs. Also, songs come with very rich metadata these days that models can take advantage of. LightFM is great to address the above two items as their method unites the advantages of content based and collaborative recommenders.

The target variable (song listen counts in this case) is expressed as a function of the dot product of the user and item latent representations, adjusted by user and item bias:

$$r_{ui} = f\left(q^u \cdot p^i + b^u + b^i\right)$$

and the latent representation for an item p^i is given by the sum of its features' latent vectors and similarly for each user q^u. This was taken from the LightFM paper by Maciej Kula.

So this model not only allows us to make recommendations just using user, item and ratings (implicit/explicit) but also can incorporate user and item attributes – this is especially useful for the cold-start problems.

Objective and Attributes:

Our objective with this method was to evaluate the model on the song listen dataset with and without using the song attributes available. The two song attributes we used here are the Artist and the Genre information. For this analysis, we only used songs that had the Artist and Genre information available.

The metrics we used for model evaluation were focused on testing the quality of recommendations from the model. For this model, we were not focused on predicting the listen count for each song for every user accurately. So, metrics like RMSE and MAE were not used. Below are the metrics:

Mean Reciprocal Rank:

The reciprocal rank is calculated as: 1 / the rank of the highest ranked positive example. A perfect score is 1.0. For example, if 10 recommendations are made and Rank 5 and Rank 7 songs were actually listened to by the user, then Reciprocal Rank is calculated as 1/5 = 0.2. This is calculated for each user and then averaged.

Mean AUC Score:

It is the measure the ROC AUC metric for a model: the probability that a randomly chosen positive example has a higher score than a randomly chosen negative example. A perfect score is 1.0. This AUC score is calculated for each user and then averaged.

Total N-Precision at k:

The number of known positives in the first k positions of the ranked list of prediction results for each user. A perfect scenario is when all the predictions/recommendations are songs that have been listened to. This is calculated for each user for a fixed k (for our analysis, we chose k=10) and then added together. This measure is mainly used as a comparison across different methods and parameters. Since this is a non-normalized metric, this cannot be evaluated in isolation. But qualitatively, this metric is easily understood – it indicates how many of our predictions were actually listened to by users.

Baseline Metrics:

Since the metrics chosen above are suited for checking the performance of recommendations and are not standard metrics used in Regression like RMSE, MAE, we had to come up with our own implementations for the baseline. While we thought of a few, we stuck to a simple baseline measure. Intuitively, if we do not have the ability to build advanced models, we might recommend the top items (most consumed songs) based on the training set to all users. While the recommendations would be the same across all users, this will be a safe bet as we recommend highly popular songs to the users. We compute both the **Total N-Precision at K** metric and the **Mean Reciprocal Rank** metric for the small and large dataset we use for our findings. More information about the small and large dataset is provided in the following sections. A quick summary of the baseline performance:

Dataset Size	Total N-Precision at k=10	Mean Reciprocal Rank
Small	3031	0.099
Large	3070	0.10

The baseline performance will be compared to the different models we discuss in the below sections

Data and Findings:

Small Dataset:

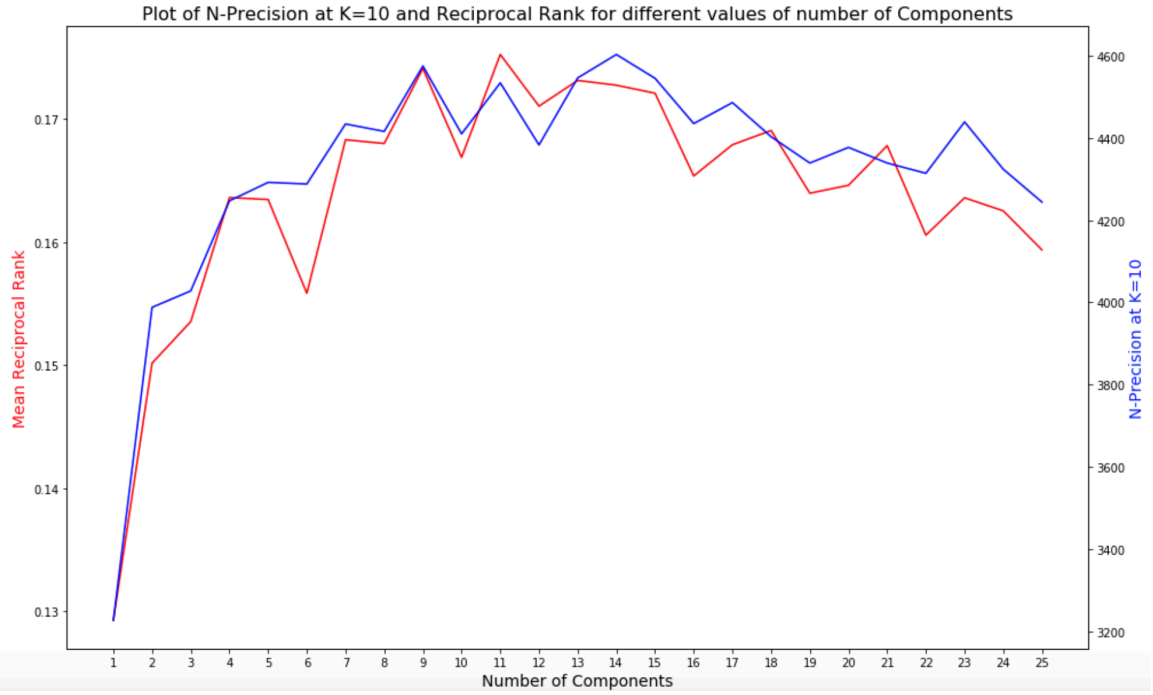
Dataset: ~10K users; 300 songs

We initially ran the algorithm for ~10K users and 300 songs. We began by just using the user, item and listen count information. For the first run, we ignored the genre and artist metadata.

user_id	song_id	listen_count
76235885b32c4e8c82760c340dc54f9b608d7d7e	SOUKXIN12A8C133C7F	3
1d04a4f845e6028b133cdb00afac4ab9c0bf422e	SOUKXIN12A8C133C7F	2
1ed24d0082434d4147ae3831d948a9b7738123ae	SOUKXIN12A8C133C7F	6
63abe7cbc1d2c45ba385e1332ec07e735269aed7	SOUKXIN12A8C133C7F	1
5ed13deffe791a7b2109e1d56ea847b99b6dd4af	SOUKXIN12A8C133C7F	2

The data was split into Train, Validation and Test Dataset. Our main hyperparameter of interest was the number of components in the learned latent representation for the users and items. We varied this parameter from 1 to 25 and used the validation dataset to decide on the best value for number of components. We plotted both the Mean Reciprocal Rank and the Total N-Precision at k=10 to decide on the optimum value. The Total N-Precision was given more importance as that was an indicator of how many of our recommendations were consumed as part of the test dataset.

Performance plot on the Validation Dataset:



On the test dataset, this model achieved:

Number of listened predictions across all users: **5404**
Mean reciprocal rank across all users: **0.189**
Mean AUC score across all users: **0.772**

Observations:

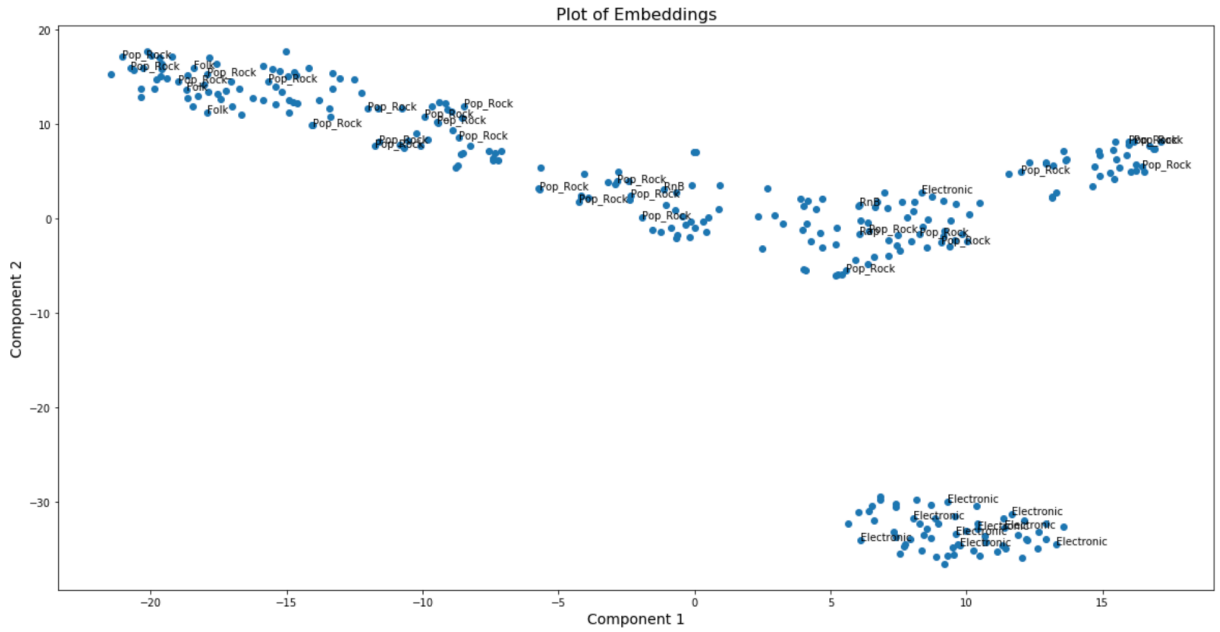
- We notice that the AUC is close to 0.8 which indicates that the model ranks positive items higher than negative items for users - from a ranking standpoint, this is good.
- The reciprocal rank is ~0.2. While that might sound low, we have a lot of items to recommend to the user and narrowing down such that we have a listened song in the top 5 recommendations made is pretty good.
- The number of listened predictions of 5404 is more useful in the context of comparison with other models.
- But it's clear that the performance of the model based on the reciprocal rank and the number of listened predictions is much higher than the baseline

Since the methodology involves embedding the items in low dimensional space, we wanted to visualize the data. We used TSNE as a dimensionality reduction technique to reduce the dimensionality from 9 to 2 for easy visualization. The plot below has every item's embedding and some random songs have been labeled using their genre information.

Before we look at the plot, let's take a look at the genre distribution.

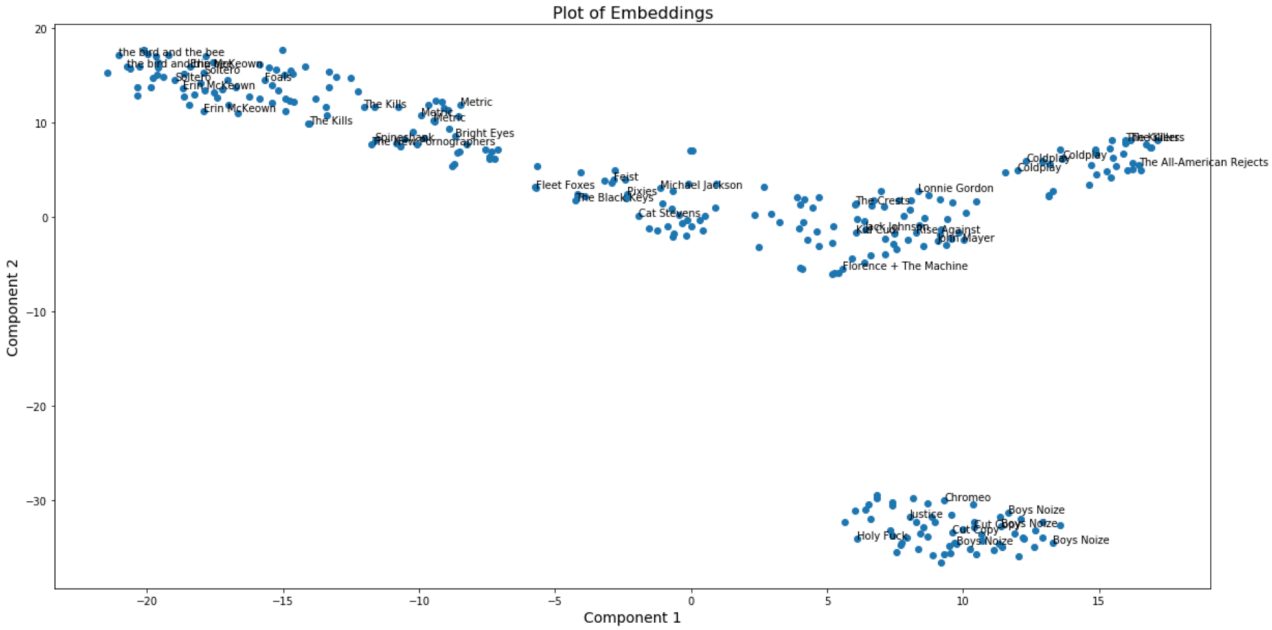
Pop_Rock	226
Electronic	43
Folk	10
RnB	8
Rap	5
Country	3
Reggae	2
Comedy_Spoken	1
Jazz	1
Latin	1

We can see that the genre is dominated by pop_rock which isn't great - we would have liked to have it split out by pop and rock ideally but let's look at how they get embedded.



We haven't fed any side information (genre) to the model. Despite that, we can see that the item embeddings cluster around the genre - electronic songs have clustered together while pop_rock songs are more spread out because they cover a wide array. This plot would look better if we had more granular genre information but this is a very interesting observation.

Similarly, we wanted to visualize and label the artist information.

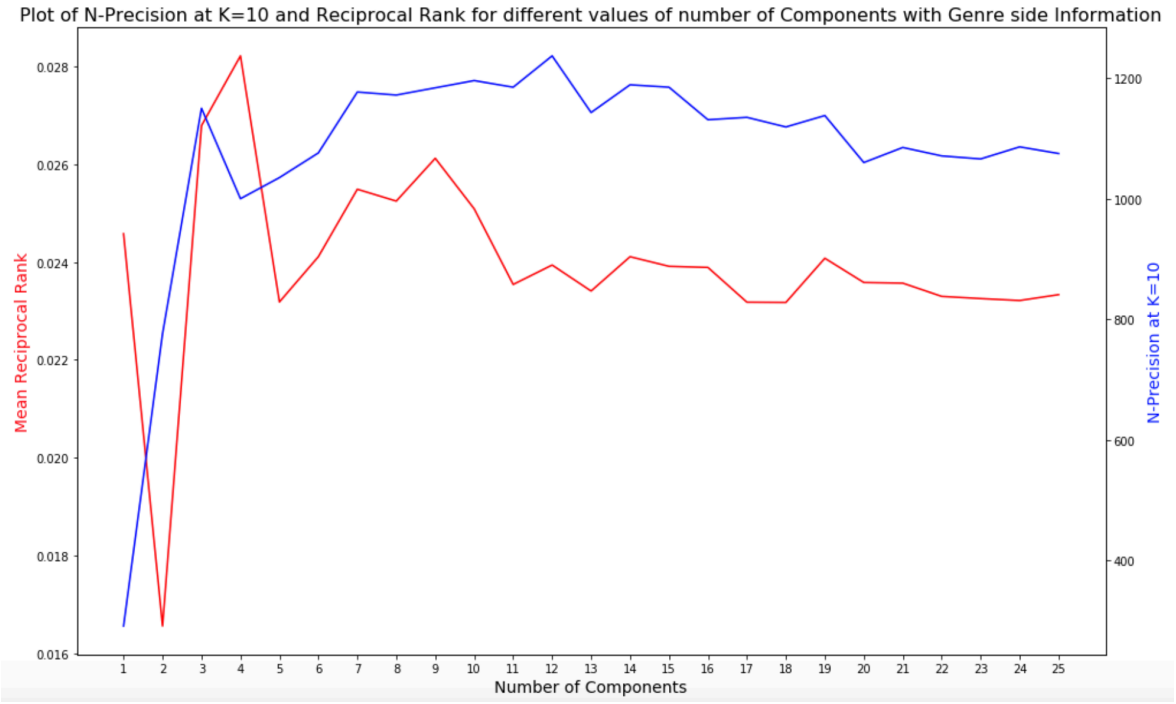


It's very clear from the plot that songs from the same artist are very close to each other in the embedding space. This is expected but is interesting to see. Also, it's good to see that Coldplay songs are close to the songs by The Killers - again, that is intuitive to us

Now, we want to add side information and check if that improves the model performance.

Adding Genre:

Performance plot on the Validation Dataset:



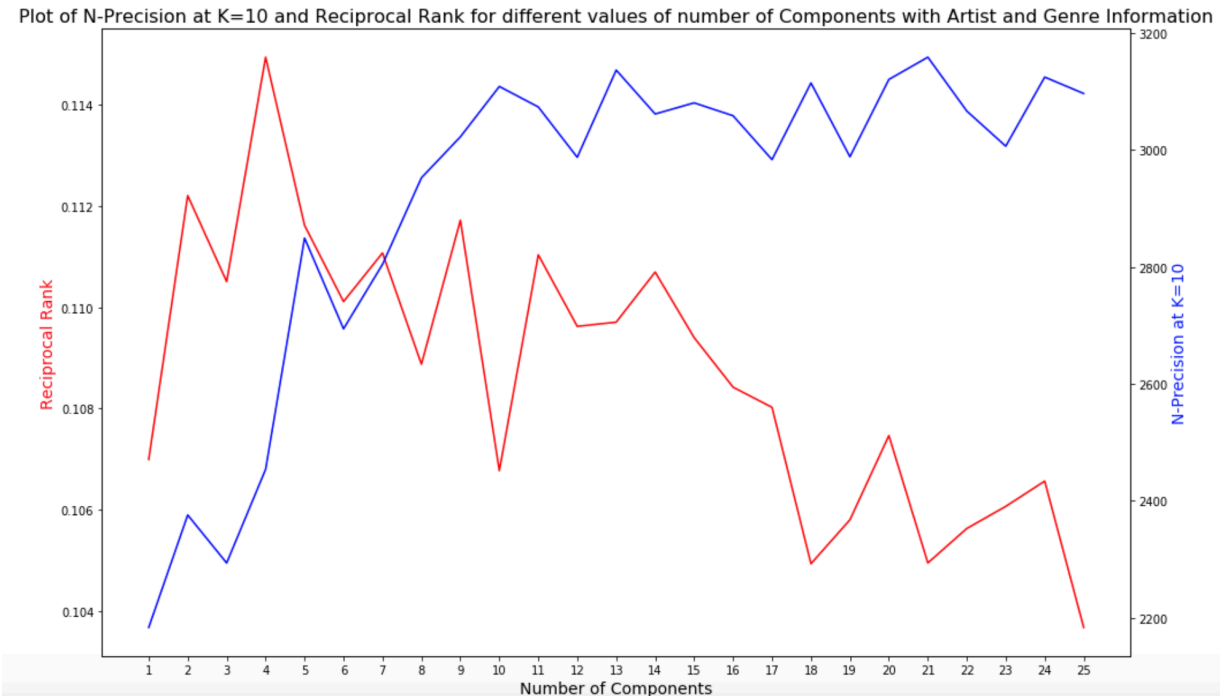
It's clear that the addition of genre made the model worse. This could be because the genre is not very granular – pop_rock convers most of the songs in our dataset. Better genre classification might yield a more useful model.

On the test dataset, this model achieved:

Number of listened predictions across all users: **1534**
Mean reciprocal rank across all users: **0.027**
Mean AUC score across all users: **0.265**

Adding both Genre and Artist:

Performance plot on the Validation Dataset:



The performance with both genre and artist information is better than the model with just genre information but it is still not as good as the performance without side information

On the test dataset, this model achieved:

Number of listened predictions across all users: **4061**
Mean reciprocal rank across all users: **0.122**
Mean AUC score across all users: **0.776**

Sample Recommendation:

Recommendation from the model without side information for a randomly chosen user:

```
Listened Songs:
Artist: John Mayer & Song: Heartbreak Warfare
Artist: Katy Perry & Song: Lost
Artist: MSTRKRFT & Song: Street Justice
Artist: Florence + The Machine & Song: You've Got The Love
Artist: Coldplay & Song: The Scientist

Recommended Songs:
Artist: Miley Cyrus & Song: Party In The U.S.A.
Artist: OneRepublic & Song: Secrets
Artist: Kings Of Leon & Song: Revelry
Artist: Coldplay & Song: Clocks
Artist: Florence + The Machine & Song: Cosmic Love
```

We see that the recommendations are pretty interesting. There is a song from Coldplay which is an artist the user already listens to and there are other songs that also seem to be similar to songs the user listened to.

Recommendation from the model with side information:

```
Listened Songs:
Artist: John Mayer & Song: Heartbreak Warfare
Artist: Katy Perry & Song: Lost
Artist: MSTRKRFT & Song: Street Justice
Artist: Florence + The Machine & Song: You've Got The Love
Artist: Coldplay & Song: The Scientist

Recommended Songs:
Artist: La Roux & Song: Bulletproof
Artist: Muse & Song: Resistance
Artist: Muse & Song: Uprising
Artist: Miley Cyrus & Song: Party In The U.S.A.
Artist: Five Iron Frenzy & Song: Canada
```

We see that the recommendations are quite different from the ones we had with the model without side information. However, they are still relevant to the user

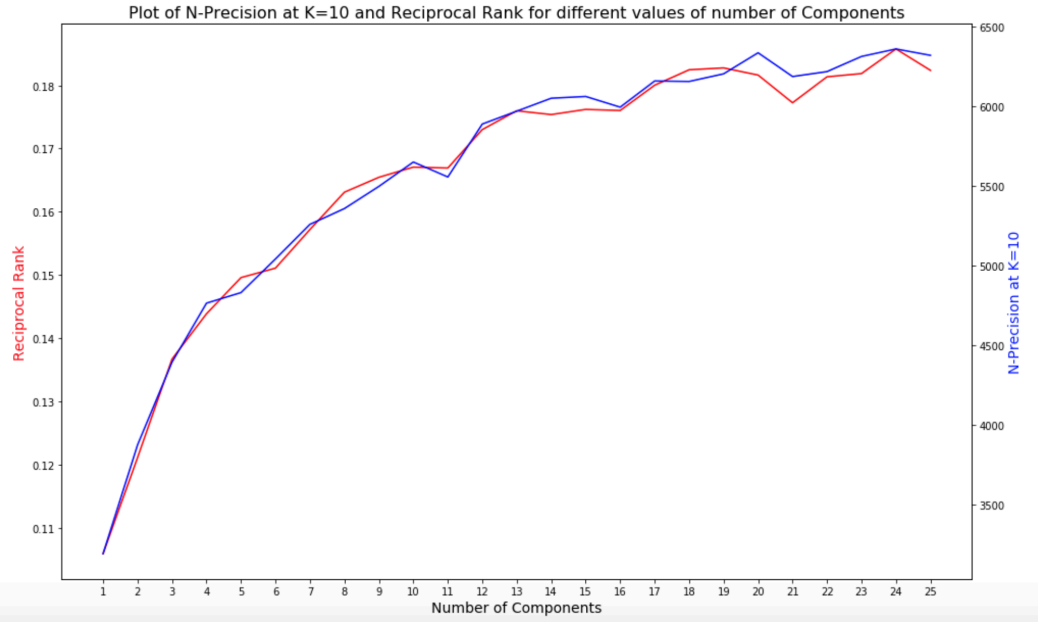
Large Dataset:

We wanted to explore the effect of increasing the size of the data on the model performance.

Dataset Size: ~10K users; 2800 songs

Model with no side information:

Performance plot on the Validation Dataset:



The graph looks much cleaner on the validation dataset with larger input data. Also, as we increase the number of components, the performance is increasing.

On the test dataset, this model achieved:

Number of listened predictions across all users: **7770**
Mean reciprocal rank across all users: **0.212**
Mean AUC score across all users: **0.843**

This is a significant improvement in performance even on the test dataset. While training on the larger dataset took longer, the performance was better.

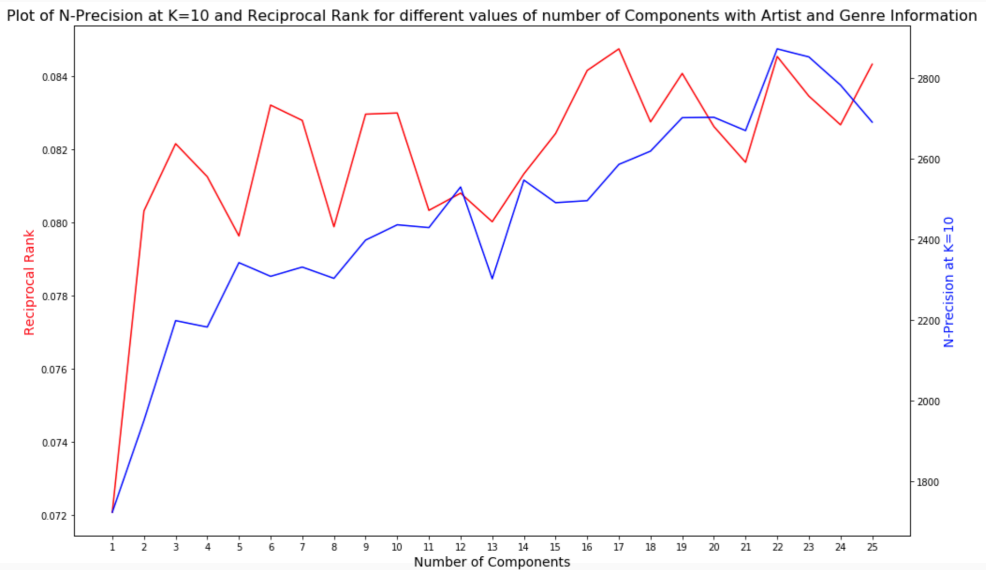
Recommendation from the model without side information for the same user:

```
Listened Songs (from Train Dataset):
Artist: Dwight Yoakam & Song: You're The One
Artist: Subhumans & Song: Rain
Artist: John Mayer & Song: Heartbreak Warfare
Artist: Junior Boys & Song: A Certain Association
Artist: CSS & Song: Meeting Paris Hilton (Album)

Recommended Songs:
Artist: La Roux | Song: Bulletproof
Artist: OneRepublic | Song: Secrets
Artist: Florence + The Machine | Song: Cosmic Love
Artist: Passion Pit | Song: Sleepyhead
Artist: Chromeo | Song: Fancy Footwork
```

Model with Artist and Genre side information:

Performance plot on the Validation Dataset:



On the test dataset, this model achieved:

Number of listened predictions across all users: **3359**
Mean reciprocal rank across all users: **0.097**
Mean AUC score across all users: **0.842**

While the AUC score hasn't dropped, the other metrics saw a significant drop as compared to the model with no side information. So we might choose to stick to the model with no side information.

Recommendation from the model with side information for the same user:

```
Listened Songs (from Train Dataset):
Artist: Dwight Yoakam & Song: You're The One
Artist: Subhumans & Song: Rain
Artist: John Mayer & Song: Heartbreak Warfare
Artist: Junior Boys & Song: A Certain Association
Artist: CSS & Song: Meeting Paris Hilton (Album)

Recommended Songs:
Artist: La Roux | Song: Bulletproof
Artist: Five Iron Frenzy | Song: Canada
Artist: Lonnie Gordon | Song: Catch You Baby (Steve Pitron & Max Sanna Radio Edit)
Artist: Dwight Yoakam | Song: You're The One
Artist: Cosmo Vitelli | Song: Robot Soul (Radio Edit)
```

Model Summary Table:

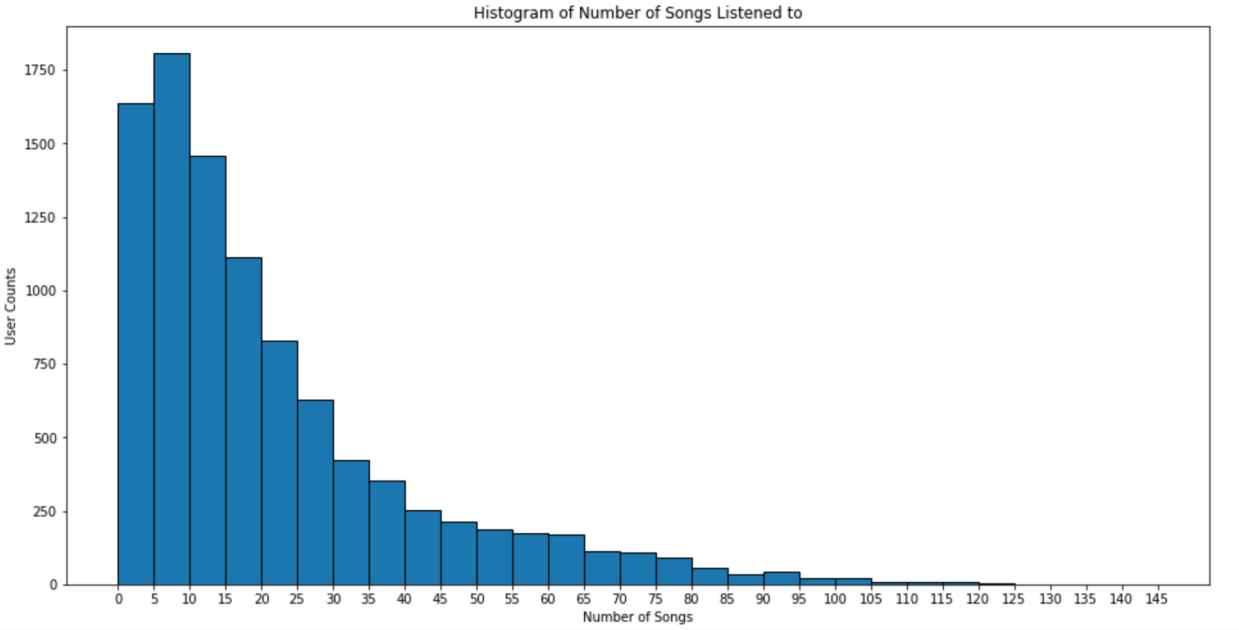
Below is the performance of various models on the small and large datasets.
Note: The metrics are computed on the test dataset.

Model	Dataset Size	Total N-Precision at k=10	Mean Reciprocal Rank	AUC Score
Baseline	Small	3031	0.099	-
No Side Info	Small	5404	0.189	0.772
With Genre	Small	1534	0.027	0.265
With Genre and Artist	Small	4061	0.122	0.776
Baseline	Large	3070	0.100	-
No Side Info	Large	7770	0.212	0.843
With Genre and Artist	Large	3359	0.097	0.842

We can clearly see from the above that our model with no side information is performing the best based on the **Total N-Precision at K** metric and the **Mean Reciprocal Rank** metric. It is also performing much better than the baseline model.

Performance on Different User Densities:

We wanted to test the model on user sets with different densities. We chose to use the smaller dataset that had ~300 songs for this exploration. Below is a histogram of the number of songs listened to by users.



We split the dataset into a sparse set which contains users who listened to less than 10 different songs and a dense set which contains users who listened more than 20 different songs. They had 3447 and 3558 users respectively. The matrices had a density of 1.6% and 14.1% respectively

We built the models with and without side information on the sparse and dense datasets.

Sparse Dataset:

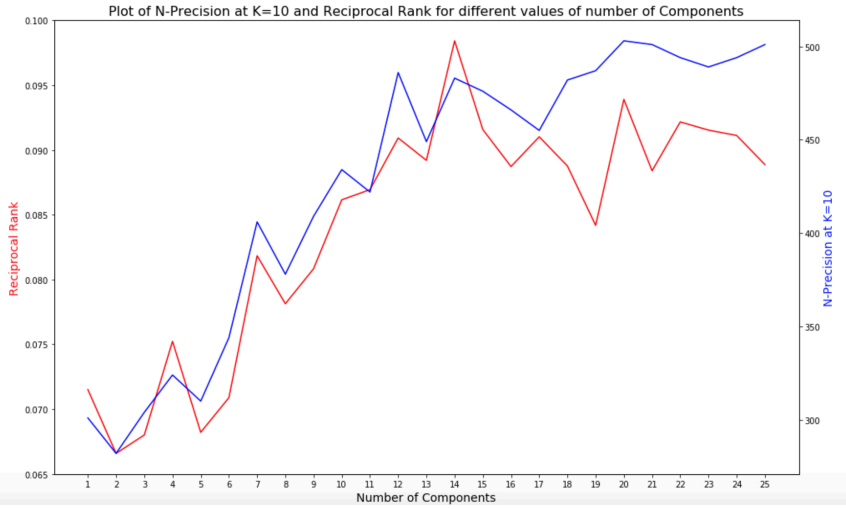
Since a subset of the data is being used, we need to recompute the baseline metrics for this dataset.

Baseline Number of listened predictions across all users: **387**
Baseline Mean reciprocal rank across all users: **0.048**

We can now compare this baseline with the models we build.

Model with no side information:

Performance plot on the Validation Dataset:



On the test dataset, this model achieved:

Number of listened predictions across all users: **627**
Mean reciprocal rank across all users: **0.10**
Mean AUC score across all users: **0.758**

We see that the number of listened predictions and reciprocal rank have dropped. The reciprocal rank is still a normalized metric and can be compared with the earlier models we built. The AUC is still pretty good and these metrics are much higher than the baselines reported for this dataset

Model with Artist and Genre side information:

Performance plot on the Validation Dataset:

On the test dataset, this model achieved:

Number of listened predictions across all users: **723**
Mean reciprocal rank across all users: **0.092**
Mean AUC score across all users: **0.765**

Unlike the other models, adding Artist and Genre information did not hurt the model. In fact, the model improved slightly on the number of listened predictions and mean AUC score. This makes us believe that in cases where the dataset is sparse, side information helps the model come up with better recommendations. This can help with cold-start problems as well.

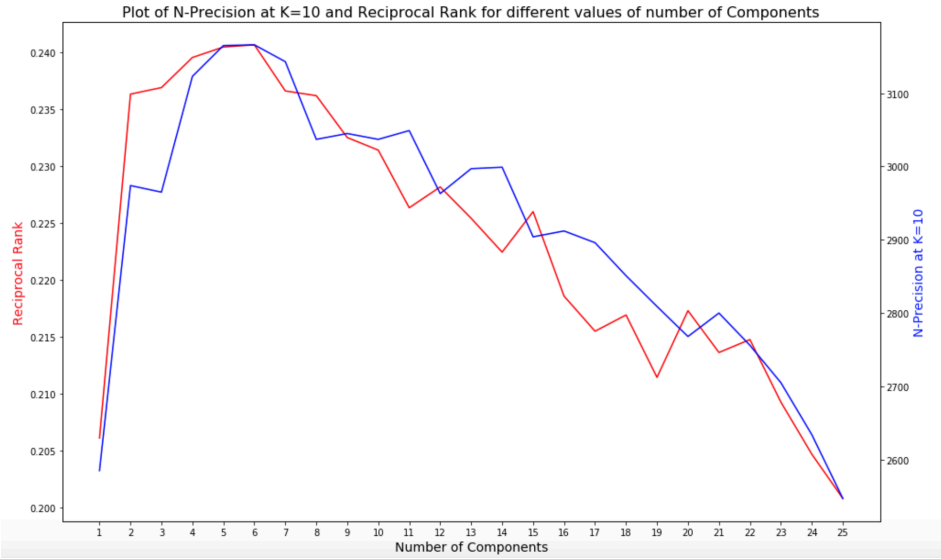
Dense Dataset:

Here are the recomputed baselines for the dense dataset

Baseline Number of listened predictions across all users: **2103**
Baseline Mean reciprocal rank across all users: **0.18**

Model with no side information:

Performance plot on the Validation Dataset:



We notice that the curve looks quite different from the curve on the sparse dataset. The optimum number of components is reached much earlier than in the sparse dataset. This indicates that the model does not require a very high dimension embedding to be learned for datasets that are dense in order to perform well.

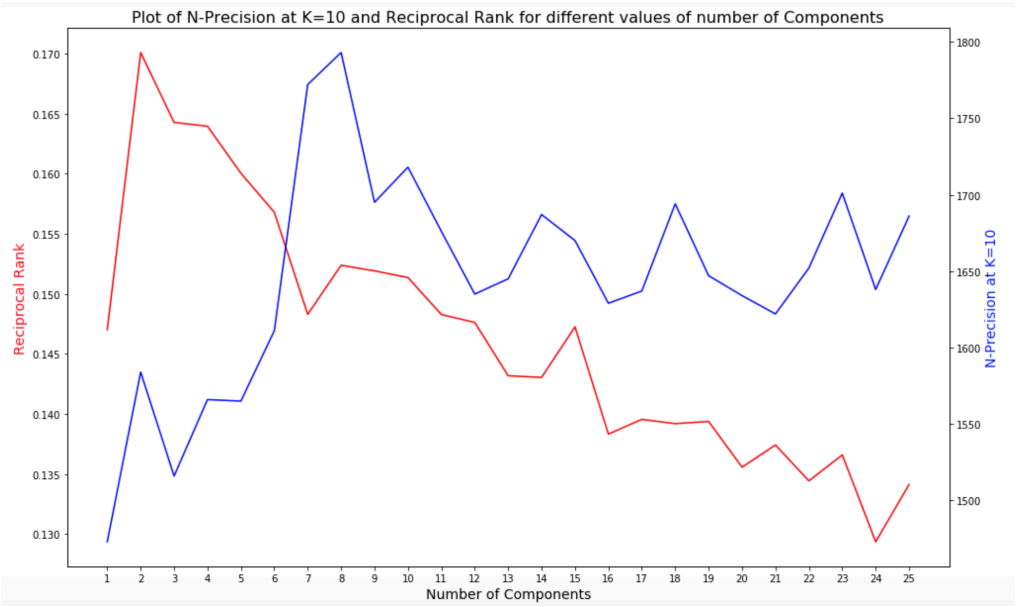
On the test dataset, this model achieved:

Number of listened predictions across all users: **3797**
Mean reciprocal rank across all users: **0.265**
Mean AUC score across all users: **0.775**

This is the best we have achieved in terms of mean reciprocal rank. In general, the model does better than the baseline like most other models we have built.

Model with Artist and Genre side information:

Performance plot on the Validation Dataset:



On the test dataset, this model achieved:

Number of listened predictions across all users: **2124**
Mean reciprocal rank across all users: **0.164**
Mean AUC score across all users: **0.773**

Again, we see a drop in the performance once the side information is added. This did not happen with the sparse dataset. We believe that the side information isn't adding much value to the model on dense users.

Model Summary on Different User Densities:

Here is the summary table containing the metrics for the above described models on user sets with different densities

Model	Dataset Type	Total N-Precision at k=10	Mean Reciprocal Rank	AUC Score
Baseline	Sparse	387	0.048	-
No Side Info	Sparse	627	0.100	0.758
With Genre and Artist	Sparse	723	0.092	0.765
Baseline	Dense	2103	0.180	-
No Side Info	Dense	3797	0.265	0.775
With Genre and Artist	Dense	2124	0.164	0.773

Next Steps on this approach:

- We need to spend more time diagnosing the reasons for the model performing poorly once side information is added for the non-sparse datasets
- We also explored Factorization Machines but ran into memory issues on large datasets. We want to compare the performance of FMs with side information compared to the Hybrid Matrix Factorization Model from LightFM
- We can add in other side information about the songs (song tags, rhythm info, etc) to see if they provide us with better recommendations.

Conclusions and Next Steps:

LSH is a good alternative to Item Based or User Based collaborative filtering when the datasets are huge and CF algorithms can't scale. As we've shown above, LSH algorithms can train in about one third of the time required for CF algorithms and can produce up to 85% similar recommendations. Please note that the results produced were run on a single node using python and this is our native implementation of LSH. LSH can be even faster with an ability to scale with hardware when we run this on a distributed cluster and leverage spark processing.

Almost most of the real-world scenarios today face the challenge of scaling up, with the ever increasing Big Data generated from all channels. LSH is a powerful and practical approximate nearest neighbors technique that can be employed to solve this problem.

Our first objective was to test out an efficient implementation of item based Collaborative Filtering using the approximate nearest neighbors method – LSH. We then wanted to pursue more modern techniques that could use user and item attributes to come up with a better performing model for recommendations.

However, when using the Hybrid Matrix Factorization Model, we realized that the model without side information such as Genre and Artist performed better than the model with side information (except on the sparse dataset) – this could be due to the lack of quality in the side information. At the same time, we do feel that the recommendations made by the model are still relevant and the model is useful in the context of a song recommendation engine.

Another approach we thought could perform well was to ensemble the two models. For each user, we could get the top K (say 10) recommendations from LSH and the Hybrid Factorization. If there are overlapping items, those are prioritized to be recommended. The non-overlapping items from the two approaches can then be alternated to get a recommendation list for the user. We plan to try this approach to check performance as opposed to using just one method or the other.

Acknowledgements:

We would like to express our sincere gratitude to prof. Brett Vintch for his invaluable support and guidance throughout the course of this project. We would also like to thank Siwen Tang for all the help and accommodating our schedules.

References:

https://towardsdatascience.com/locality-sensitive-hashing-for-music-search-f2f1940ace23

http://web.iitd.ac.in/~sumeet/Slaney2008-LSHTutorial.pdf

LightFM Paper - https://arxiv.org/pdf/1507.08439.pdfm