Last.fm New Artist Recommendation System (Final Project)

E 4571: Personalization Theory and Application

Tim Kartawijaya (tak2151), Charlene Luo (cl3788), Fernando Troeman (ft2515)

1. Objective

The goal of this project is to build a reliable recommender system that generates new and relevant artist recommendations to Last.fm users. Our system recommends the top-k (e.g. top 20) artists for each user, which the user have not interacted with in the past. Each recommendation is personalized to each user and is 'learnt' from his or her listening habits and history. The business objective of our recommender system is to increase user activity on the website and user loyalty to the Last.fm brand by providing a service that allows users to discover new artists and have a more enjoyable listening experience.

2. Data

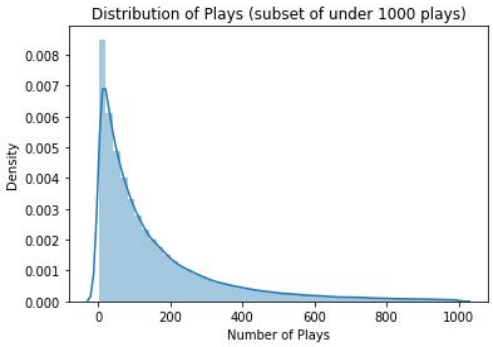
a. Source

The dataset used in this project consists of 360,000 Last.fm users with their implicit feedback of their top 50 most played artists. Each observation has the user ID, artist ID, and number of times that user listened to that artist. The data was collected from the Last.fm API by Oscar Celma, who gathered the data in the Fall of 2018 (https://goo.gl/gxPTuK). In this project, we first worked with a subset of the dataset, containing a total of 9,000 users and 47,102 artists, then increased our input size iteratively to about 150,000 users to see how well our model scales.

We also acquired metadata in the form of contextual user information, namely age, gender and country. This metadata allows us to increase the effectiveness of the models that can utilize such additional information (Factorization Machines), enhancing the quality of generated recommendations. We also attempted to gather item metadata (artist tags) using the Last.fm API (https://www.last.fm/api), but due to API limitations and the slow nature of the API calls, we were not able to gather item metadata for all our items. Our code to extract such tags can be found in artist_features.ipynb.

b. Quality

The dataset containing the user-artist interactions is ~99.09% sparse. Only a few rows in the dataset had some missing values (~1.3% of the data). Therefore, since it's only a small portion of the datset, we can drop said rows to handle the missing values. The number of plays in the dataset is distributed as such:



As shown above, the dataset is exponentially distributed since a majority of users only play artists a few hundred times. The models that we use in this project have already built-in structures to deal with this kind of data, but in case, we also set aside a scaled version of the dataset (take the log and min-max scale) so that interaction values would be only from 0 to 1.

For user metadata, we proceeded to one-hot-encode each column. In regards to missing values, this dataset has a larger amount of missing values, with ~9% of values missing for the sex feature and ~21% values missing or erroneous for the age feature. We decided to one-hot-encode these missing values as a separate category, in hopes that there is some information that the values are missing not by random.

3. Models

For this project, we employ models driven by factorization machines to generate artist recommendations for Last.fm users. Specifically, we utilized LightFM, a Python implementation of useful and relevant recommendation algorithms for both implicit and explicit feedback (https://lyst.github.io/lightfm/docs/home.html). We first use a 'vanilla' implementation of LightFM to gauge how factorization machines perform on our dataset. Next, we implement an advanced model by adding contextual data in the form of user features, incorporating user information such as gender, age and country.

For benchmark comparisons, we included the results of models that we have previously implemented:

a. Baseline: Most Popular Artists

Our "naive", or baseline model, simply recommends each user the top 20 most popular artists across all users. Top artists are obtained by ranking the total number of listens for each artist. In this case, every user receives the same recommendation. Such a model is extremely easy to build and thus we expect more advanced models to convincingly outperform this benchmark in order to be worth implementing.

b. Model-Based (ALS)

In our past attempt, we found that the Alternating Least Squares (ALS) Matrix Factorization model, implemented in the implicit package, had the best performance in regards to accuracy and training time. It is thus fitting for us to have this model as the model to beat. In this project, we will use the same hyperparameters and evaluation methods as used in the previous report.

4. Evaluation

To evaluate our models, we chose to use the holdout validation method. This method is appropriate for collaborative filtering settings, since during train-test splits, the training data will still contain user-specific data which are needed for proper training.

a. Train Test Split

First, we split our dataset into a training set and a test set on a per-user basis. For each user, we took the top-k artists that users have most listened to, then randomly sample a percentage of artists from

said top-k set to create our test set for that user, i.e. the holdout set. Then, we create the training set by updating the values of those sampled user-pair artists to zero, effectively masking and "holding-out" data from training. To evaluate these models, we can then compare the holdout set with recommended items, since we would expect the holdout items to be included in the set of recommendations. For LightFM, we employ the same methodology but implement it using LightFm's built-in cross validation method (https://lyst.github.io/lightfm/docs/cross_validation.html) in order for the datasets to be

b. Metrics

We chose to use recall at k and precision at k as our model accuracy metrics. We determine recall by calculating the ratio between the number of times a holdout item appears in the recommendations with the total number of holdout items (True Positives / Holdout Set Items). For precision, we calculate the metric by calculating the ratio between the number of holdout items that appear in the recommendations with the total number of items recommended (True Positives / k-items recommended). In our past attempt, we also used NDCG as a metric, but since LightFM does not support NDCG as an evaluation metric we decided not to use it. Furthermore, our attempt at calculating NDCG manually did not turn out as expected, as the results did not make sense (much lower than recall @ k). We opted not to use error metrics like RMSE and MAE and precision-based metrics, since these metrics are not insightful for implicit data and top-k settings.

In addition to accuracy metrics, we also used catalog coverage as a metric of the diversity of our recommendation. A catalog coverage of 20% means that out of the 47,102 artists in our dataset, only 20% of them appear in at least one user's top-20 recommendations. We also evaluate our models' serendipity by qualitatively determining whether "new users" that we create receive recommendations that are serendipitous. We decided not use novelty as a metric since we have no information how our data values relate to time.

5. User Groups

Different recommender systems work best for different types of users. To explore which algorithms are ideal for datasets of a specific nature, we grouped users up into several groups based on 3 distinct criteria:

a. Activity

We would reasonably expect algorithms to work differently for users whom which we have lots of data points to work with. As such, active users with a greater number of artist plays should reasonably expect to receive better recommendations. On the contrary, the tastes of users with low activity are harder to gauge, and thus most recommender systems will not work as well for these users. We measure activity by the number of total listens each user has logged through Last.fm.

b. Diversity

Music listening habits differ greatly across users - some mostly listen to a select few of his or her favorite artists, while others exhibit artist plays that are more evenly distributed across numerous artists and genres. It is interesting to see how our algorithms perform for users with varying levels of diversity in their listening habits.

We measure diversity by attaching a diversity score to each user, which is calculated by the spread of listens across all artists played. The artist plays for each user is scaled to between 0 and 1, where 1 is attached to the artist with the most number of plays for that user. The score is then computed by taking the sum of these scaled values. This means that users who have listened to more artists generally have a higher diversity score. Furthermore, it also means that users whose artist plays are more evenly spread out have a higher diversity score as well. For example, a user with [500, 1, 1, 1] listens will have a lower score than a user with [200, 50, 150, 125] listens.

c. Mainstream

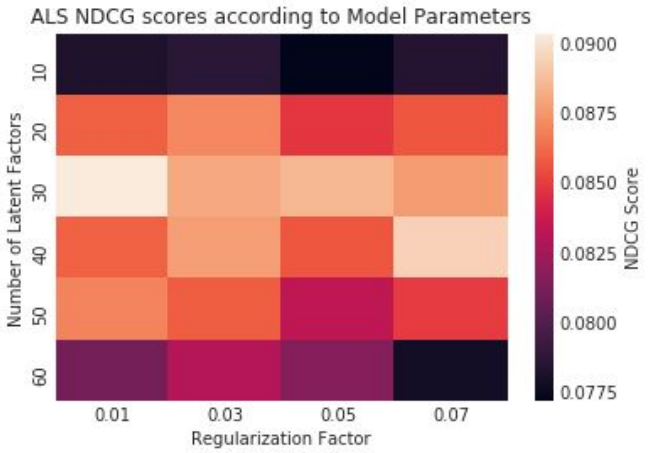
An ideal recommender system does not alienate users with niche preferences. Mainstream music is evidently more popular, and mainstream artists generate more plays from a greater number of users. As such, a good way to evaluate the quality of any recommendation algorithm is to see how it works for users with more indie inclinations.

We measure how 'mainstream' each user is by similarly attaching to them a score, which is determined by their weighted plays of popular artists. We first compute the popularity of each artist by calculating their respective total plays across all users. Following which, we assign an indicator for how mainstream a user is by computing the weighted total of listens to the top n artists. n is a parameter that we can tune to adjust which artists are considered 'popular'.

6. Parameter Tuning

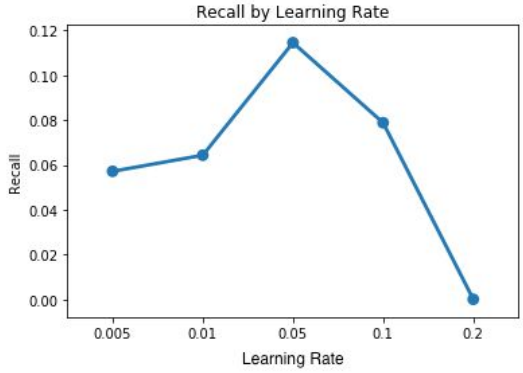
To optimize the accuracy of these models, we tuned the parameters of each model to maximize recall. To do this, we held off a test set from the data and split the rest into k-fold training and tuning sets. This way, we were able to fit a range of parameters on each training set and select the parameters that produced the best recall on the respective tuning set. With k-folds, we were able to perform this k times.

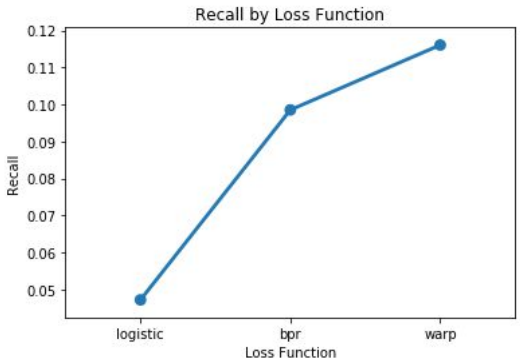
For ALS Matrix Factorization, there were two factors to tune: the number of latent factors and the regularization factor. The number of latent factors determine the number of features/attributes that describe a user or an artist upon factorization. The regularization factor is a constant in the minimization function of this algorithm that makes sure there is no overfitting in the model. We show the results we found from the last assignment, where we found hyperparameters to maximize NDCG, which is directly related to recall. The heatmap of the parameter tuning is shown below, where the optimal parameter is **30** for latent factors and **0.01** for regularization.

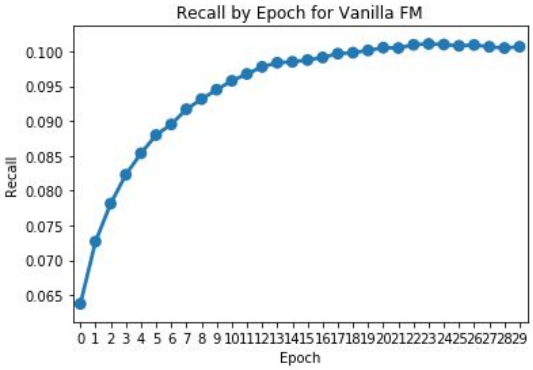


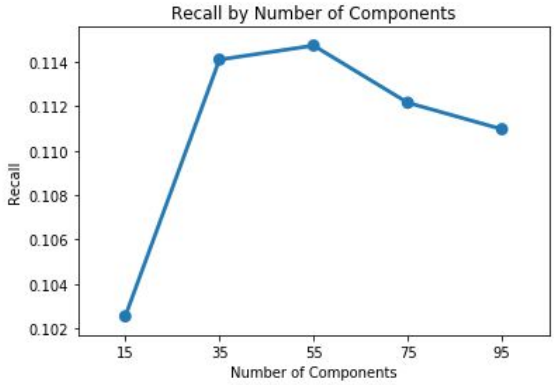
For LightFM, we have a few more hyper parameters to tune, in which we focused on three to optimize recall: number of components, learning rate, and loss function. Number of components is the dimensionality of the feature latent embeddings that we are to calculate using lightFM, learning rate is the step size of the stochastic gradient descent that learns these embeddings, and finally the loss functions consist of three functions: logistic, bpr, and warp. From Light FM documentation, it is said that BPR is supposedly best for optimizing recall, but we decided to implement all three to see how the functions perform in practice. The results are shown on the graphs below, where we find the optimal number of components to be about **50**, learning rate to be **0.05,** and the loss function to be **WARP.**

Furthermore, we found the optimal the number of epochs to run for this model while troubleshooting the model, where the recall stabilizes after epoch = 25. The progress of recall by epoch can be seen in the graph below. Understanding what the optimal epoch is enables us to be more efficient and accurate during the training of the model.









7. Results

a. Overall Accuracy

After training and testing the models, we arrive to the conclusion that the ALS method is superior in terms of recall, the Vanilla LightFM does best in regards to precision and coverage, while the additional user information did not improve the model at all. ALS model had the greatest recall rate, but had much lower coverage rates than Vanilla FM and FM with User Features. Given that our objective is to expand the scope of our users' music preferences, we should seek to improve the coverage of our ALS model. Looking at FM with User Features, the model has a worse recall even compared to the baseline, at around 3.25%. From these experiments, we state ALS as the best model, since it clearly has the highest recall rate, and will likely create the best experience for users.

Model	**Precision**	**Recall**	**Coverage**
Baseline	6.91%	6.64%	0.04%
ALS	3.26%	**21.74%**	8.53%
Vanilla FM	**14.17%**	9.78%	**27.6%**
FM with User Features	4.72%	3.25%	13.1%

Our results have shown that the LightFM model did not work as well as initially expected. An attempt to understand the cause of this underperformance has led to several plausible explanations. For instance, the inherent nature of our dataset may not be conducive for a LightFM-driven model. The Last.fm data set used in this project is extremely sparse, consisting of at most the top 50 played artist for each user. We would reasonably expect the recommendations made by LightFM to be more accurate and diverse given a wider range of data points to learn from.

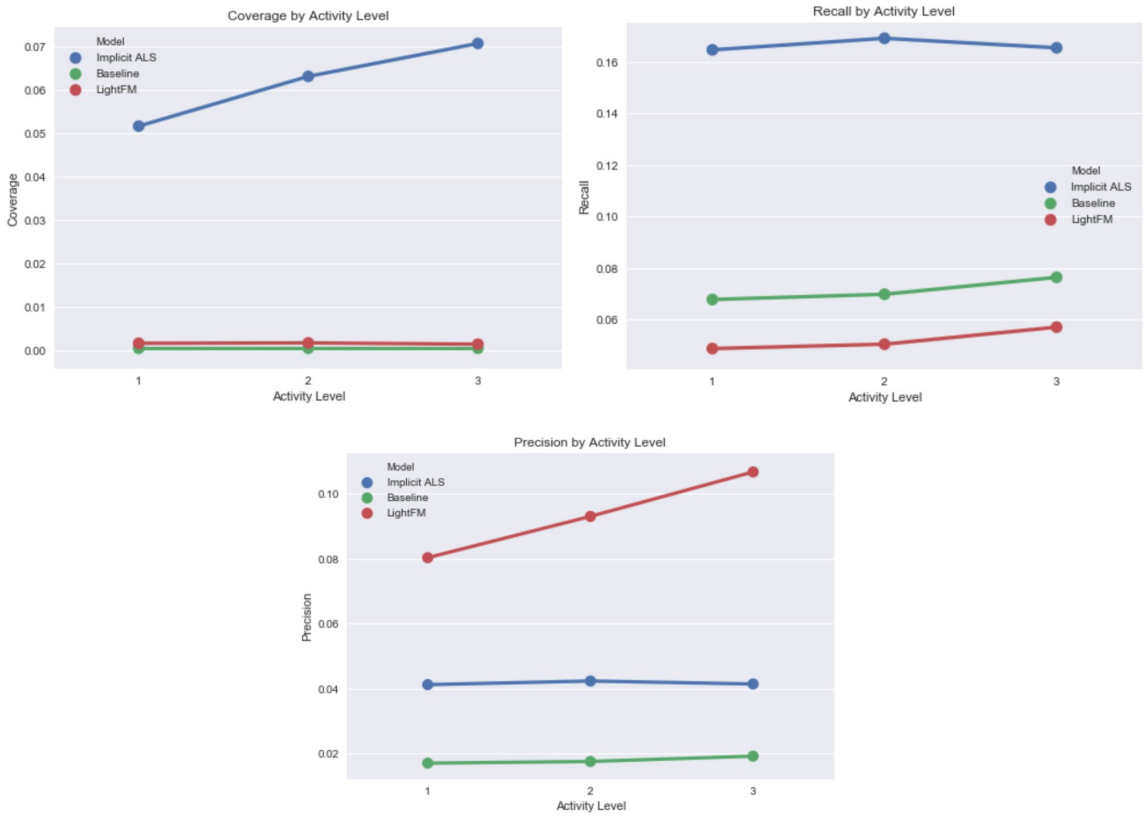
The addition of user features also led to a worsening of the model's performance. This could possibly mean that gender, age and country each have a negligible effect on any given user's listening habits. As such, the addition of this metadata only contributed to the noise present in the dataset, worsening the accuracy of our recommendations. The quality of the metadata available to us is also questionable, with numerous missing values all around as well as unrealistic values, especially for age.

These are just some plausible justifications as to why a model driven by ALS worked better in our case as compared to models driven by factorization machines through LightFM. Although LightFM is conventionally regarded as a more advanced model, the dataset in question has to be suitable for its use.

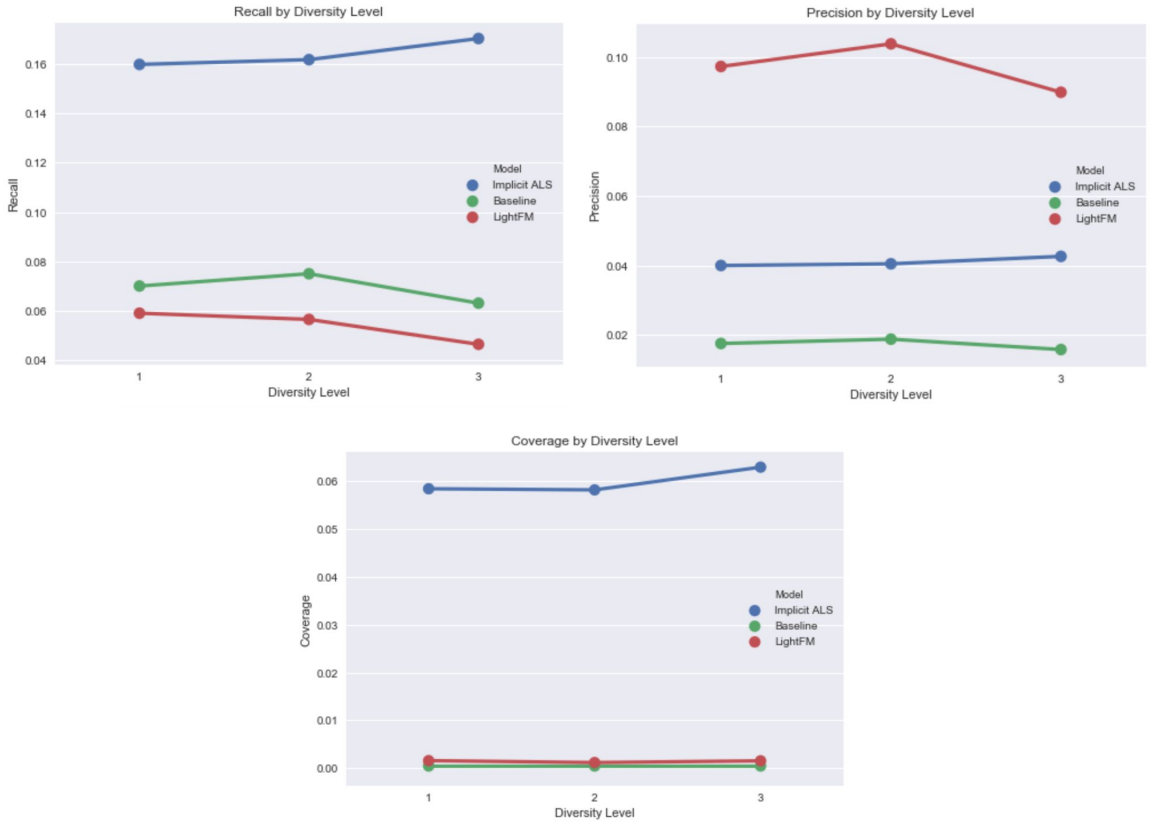
b. Accuracy Results by User Group

As described in Sections (5.a-c), we split the users into groupings along three different scales: activity level, listening diversity, and popularity of music listened. Overall, LightFM (with User Features) performed poorly in recall but well in precision; Implicit ALS had the best recall; Mainstreamness of music taste has the largest impact on recommendation quality.

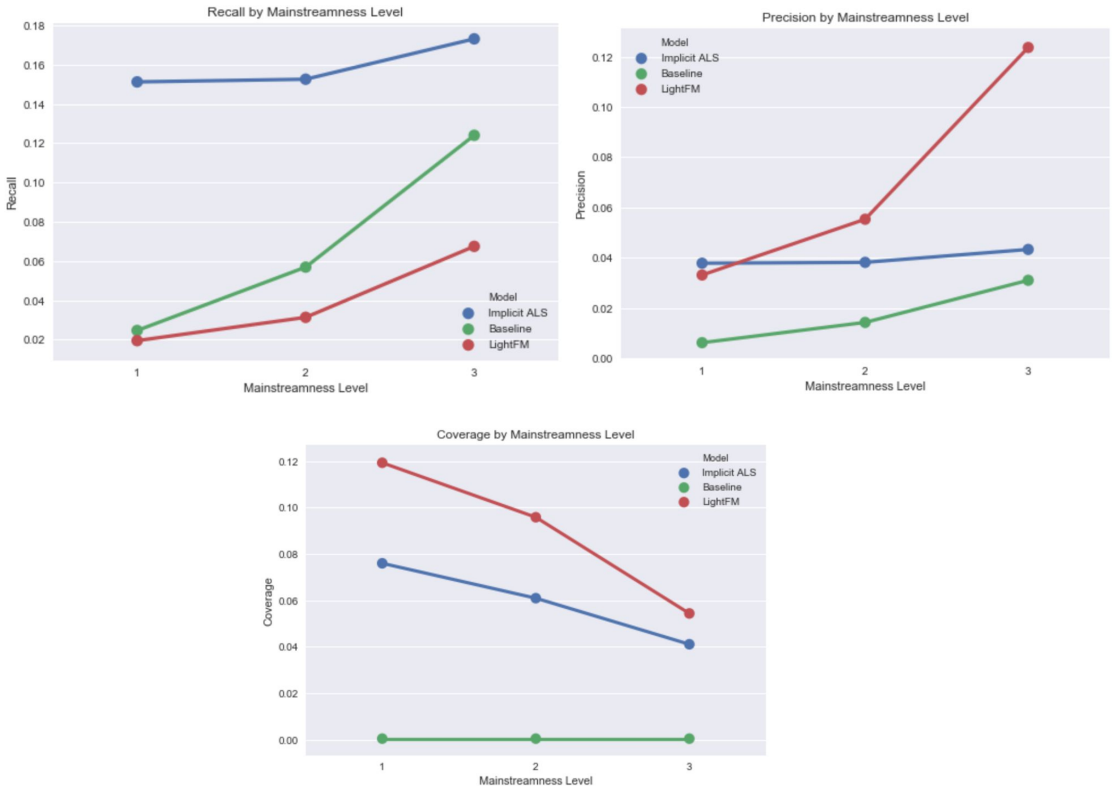
When users were grouped by activity level, there did not seem to be a very clear pattern from recall and precision levels. However, for LightFM, there seems to be a increasing trend for precision as activity levels increase. In these graphs, activity levels 1, 2, 3 are encoded in the same way as described before: with three levels, level 1 includes a group of the bottom 33% of users in terms of activity, level 2 is the middle 33%, and level 3 is the top 33%. LightFM produced disappointing recall scores, lower than even baseline. LightFM did not perform well in recall for all three groupings.



In these graphs, diversity levels are again determined in the same percentile increments as in activity levels (by 33%). When users were split by diversity of listening activity, LightFM again had the best precision while it had the lowest recall and coverage. This is probably because in our experiments, LightFM tends to recommend similar songs to all users, similar to the result from baseline. Overall, there seems to not be a very clear relationship between quality of recommendations and the diversity level of a user's music taste.



Of the three scales, the popularity of music listened, or ‘Mainstreamness”, showed the most pattern in recall, precision, and coverage. In these graphs, the “Mainstream” Levels are defined in the same 33% percentile increments in “mainstream” levels as described earlier. As shown below, LightFM does not perform very well with recall, but it is very precise and has decent coverage compared to Implicit ALS and Baseline. These graphs also make intuitive sense: as users are more mainstream, the coverage decreases since recommendations are more narrowed around popular artists, and the quality of recommendations goes up since these users are more predictable.From these graphs, we can conclude that mainstream users will receive markedly higher quality (due to higher recall and higher precision) recommendations than those with very alternative music tastes.



c. Qualitative and Serendipity

We now qualitatively evaluate how our models perform in giving recommendations. First, it is important to note the most popular artists in this dataset. We determine the most popular artists from using the baseline model: most popular artists. From the results below, we can see that there is a significant number of rock/alternative rock artists (e.g. red hot chilli peppers, pink floyd), indicating to us that the group we sampled from listens to older music. This is reasonable since the dataset is extracted on 2008.

```
['martingo',
 'the beatles',
 'radiohead',
 'madonna',
 'coldplay',
 'nine inch nails',
 'marilyn manson',
 'metallica',
 'muse',
 'pink floyd',
 'linkin park',
 'red hot chili peppers',
 'system of a down',
 'death cab for cutie',
 'placebo',
 'nightwish',
 'depeche mode',
 '2pac',
 'the killers',
 'in flames']
```

In order to understand how accurate and serendipitous our system's recommendations are, we create "fake" user profiles that represent different various user segments, then insert these new users into the same dataset that we train and evaluate on. Then we get recommendations for these made-up user profiles, and qualitatively assess how good these recommendations are. We included our own music tastes to determine the serendipity of the recommendations. The table below lists the users we created, along with what segment that they represent and a list of artists that they listen to. As a default, we set each artist-user pair to 1000 plays to indicate highly positive feedback.

User Name	Segment	Artists
Rap	Non-Diverse, Non-Mainstream (rock is mainstream at this point)	'kanye west', '2pac', 'lil wayne', 'eminem', 'young jeezy', 'jay-z', 'drake'
Rock	Non-Diverse, Mainstream	'the beatles', 'the rolling stones', 'led zeppelin', 'queen', 'pink floyd', 'ac/dc', 'guns n\' roses', 'aerosmith'
Tim	Non-Diverse, Non-Mainstream	'daft punk', 'deadmau5', 'john mayer', 'hans zimmer', 'coldplay'
Charlene	Diverse, Semi-Mainstream	'solange', 'sufjan stevens', 'beirut', 'yo la tengo', 'little dragon', 'crystal castles', 'a tribe called quest', 'radiohead', 'faye wong','the beach boys', 'van morrison', 'marvin gaye', 'whitney houston', 'kanye west', 'curtis mayfield', '2pac', 'eagles', 'david bowie', 'prince'

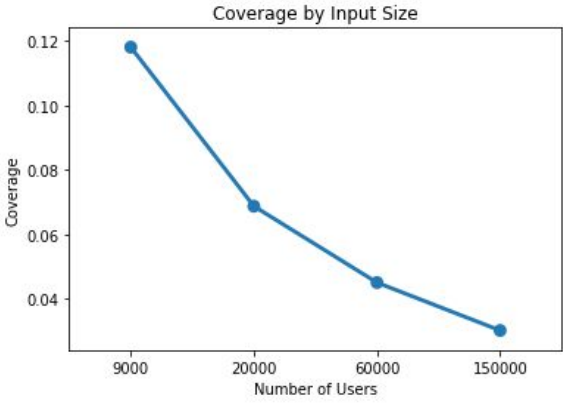
For each of these users, we then get recommendations from both the Implicit ALS model and the LightFM model with user side information. The table below summarizes the recommendations given by the LightFM with user features model and the Implicit ALS model. For the sake of space, we will include only Rap and Tim in this report. The full results can be found in section 4c at Main.ipynb

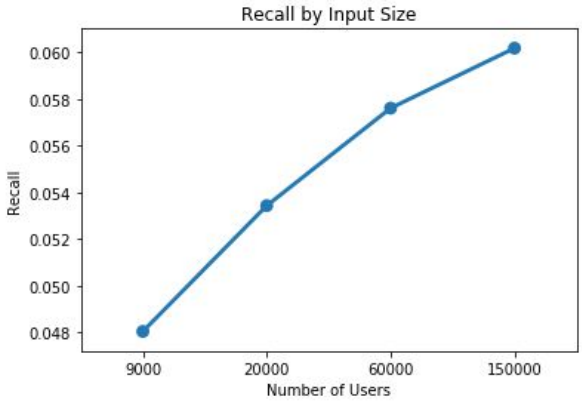
	Implicit ALS	**LightFM**
Rap	't.i.', '50 cent', 'ludacris', 'lupe fiasco', 'akon', 'snoop dogg', 'the game', 'nas', 'notorious b.i.g.', 'chamillionaire', 'timbaland', 'nelly', 'bone thugs-n-harmony', 'dmx', 'dr. dre', 'justin timberlake', 't-pain', 'flo rida', 'usher', 'ice cube'	'the beatles', 'coldplay', 'bloc party', 'jack johnson', 'jay lumen', 'radiohead', 'the grand silent system', 'cat power', 'incubus', 'the smashing pumpkins', 'the strokes', 'linkin park', 'vampire weekend', 'red hot chili peppers', 'sigur r\xc3\xb3s', 'the promise ring', 'hot chip', 'feist', 'the white stripes', 'air'
Tim	‘Martingo’,’coldplay’,’michael jackson’,’kanye west’,’hans zimmer’,’john mayer’, ‘soundtrack’,’linkin park’, ‘red hot chili peppers’, ‘muse’, ‘eminem’, ‘the prodigy’, ‘the killers’, ‘moby’, the beatles’, ‘deadmau5’, ‘pendulum’, ‘david guetta’, ‘gorillaz’	'the beatles', 'the notwist', 'red hot chili peppers', 'death cab for cutie', 'coldplay', 'radiohead', 'broken social scene', 'the smashing pumpkins', 'air', 'daft punk', 'zombie girl', 'manu chao', 'jack johnson', 'placebo', 'the grand silent system', 'fair to midland', 'queens of the stone age', 'adele', 'fall out boy', 'regina spektor'

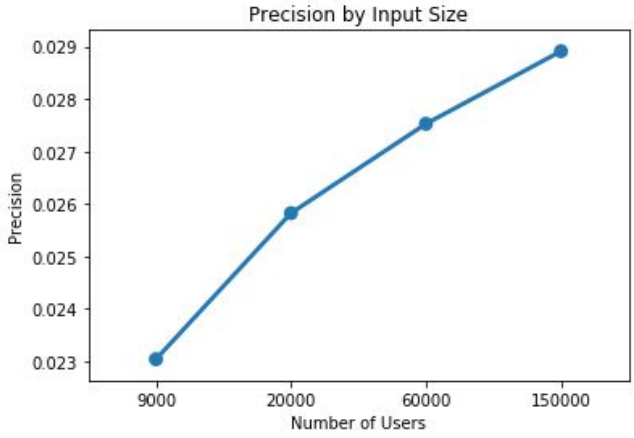
As shown in the results, we can see that for “Rock,” both models generally perform well, since the default of the models tend to recommend the most popular, which has the most data. On the other hand, for “Rap,” the ALS model performed much better, giving artists similar to the user’s taste, while LightFM defaults to rock artists. Within our own profiles, the team found that the results from ALS performed better than LightFM. For example, for Tim, he found “gorillaz” and “moby” as new artists from Implicit that fits his taste, while in from LightFM, he found that most recommendations were metal/rock bands, which he did not enjoy. Note that some already liked items are recommended again. This might be due to a bug within our recommendation code but can be easily fixed in the future. In conclusion, qualitatively ALS performed better, giving recommendations that are similar to a user’s tastes albeit a few false positives (e.g. Tim’s eminem).

d. Scalability

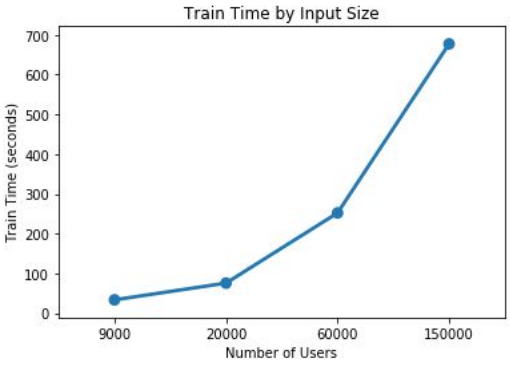
In addition to quantitative and qualitative metrics, we explore how scalable the LightFM model with user metadata is in regards to accuracy, training time, and recommendation time. We do this by increasing the input size i.e. the number of users included in the dataset and evaluate the model. From the results above, we have seen that the LightFM model isn't performing as well as we hoped, thus we wanted to see if adding more data would improve the LightFM model drastically. For reference, 100,000 users amount to about 5 million ratings. Also, note that we ran the models below on the Google Cloud Platform to run the code more efficiently, using an Ubuntu 18.04 LTS Virtual Machine with 8 vCPU and 52 GB of memory. The results are shown in the graphs below.

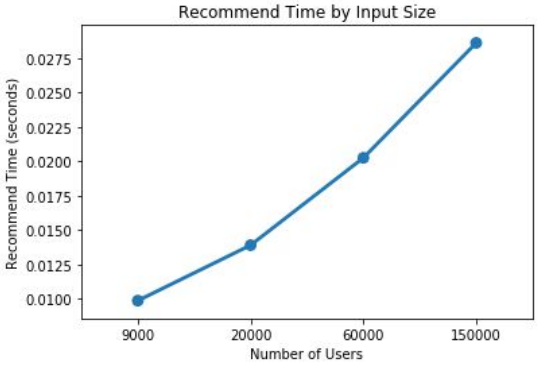




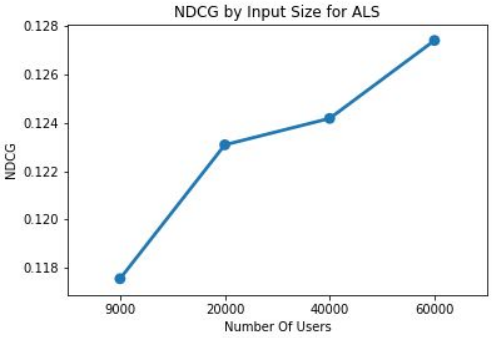


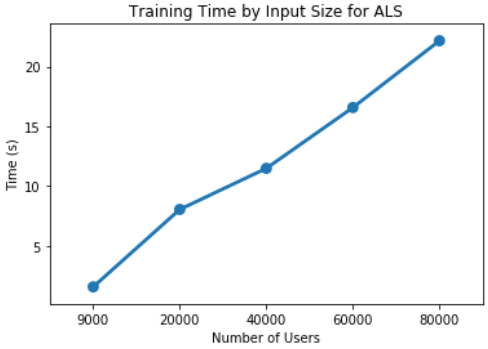
As we can see from the accuracy metrics, precision and recall increases slightly as input size grows. On the other hand, coverage suffers drastically, since the number of items/artists added grows significantly. In regards to training and recommendation time, the model does not seem to scale as well, since recommendation and training time grows exponentially as input size grows, as shown below.





LightFM with user features has shown to be scalable in regards to accuracy, yet when compared to ALS, LightFM is still not as efficient or accurate as implicit's ALS model, as shown in the ALS results below.





8. Conclusion and Next Steps

In conclusion, we found that Implicit's ALS Matrix Factorization is still the best model to provide top-k recommendations for the LastFM dataset. The LightFM model did not perform as well as we anticipated, failing to reach the recall rate of the Implicit model even without side information. We hypothesize that this is because the Implicit package is tuned and better suited for non-binary implicit data like LastFM. Furthermore, adding side information decreased the accuracy of the model, which we hypothesized to be due to the quality of the side information. Even though the LightFM model is scalable in regards to accuracy and training time, ALS still outperforms LightFM in scale. When users were split along scales of activity level, diversity of musical tastes, and "mainstreamness", "mainstreamness" proved to be the best indicator of recommendation quality. The more mainstream a user's listening activity is, the better our models were able to recommend music to them. And as was true in previous tests, ALS outperformed the other models in terms of recall while LightFM had fairly high precision. Qualitatively, ALS also is shown to provide recommendations that are better suited to a user's taste, not defaulting to popular music like LightFM.

Moving forward, we can further attempt to improve the model's accuracy metrics by gaining cleaner data in regards to user demographics and item metadata. Outside of Factorization Machines, we can try other methodologies to improve recommendations, such as association rules and MNAR.