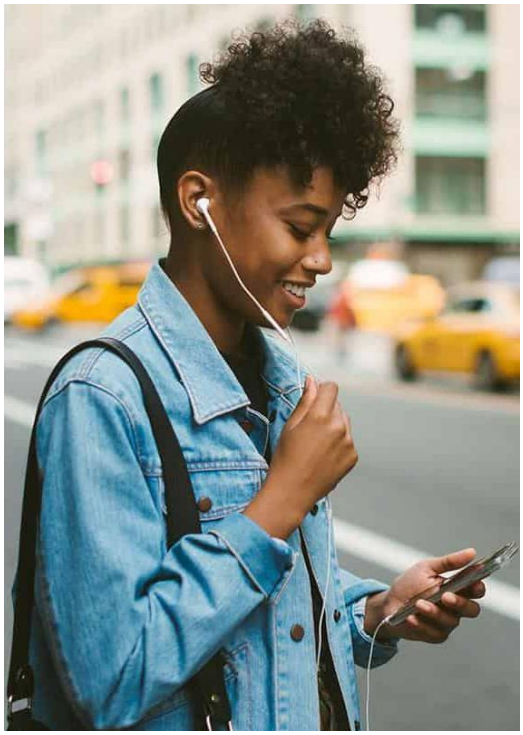
Group project

Communication tools
Philipp Borchert



Improve the recommendation system of LastFM

Fangda Fan, Mario Cortez & Sofie Ghysels

Executive summary

Due to new trends and an increasing market, it will be paramount in the music streaming service to retain your users/customers. Therefore, LastFM will have to improve their recommendation systems to provide their customers with a better experience and increase the amount of time spent on their platform.

The added value of this project for LastFM is threefold. Firstly, we provide two clean base matrices which can be easily used for further data analysis. Secondly, we have dived further into the recommendation systems than just the traditional metrics and therefore can give recommendations related to variety, more specifically how LastFM could encourage its users to engage with a variety of artists. Thirdly, our project results in a clear structured plan and sufficient information (for instance on the pros and cons of the models), which can be used by not only data scientists but also the marketing and communications department to gain better insights into the users of LastFM.

It should be noted that none of the metrics that we discussed matter more than how the customers themselves react to their recommendations. Here, the marketing and communications department of LastFM can complement the data science by standing close to the users. Actual user behavior is the test of our analysis. Therefore, we would like to point out the importance of performing A/B online tests.

Table of contents

The music streaming market

A music streaming service is a web-based service that allows users to stream songs to their computers or mobile devices. Users can also download songs to play them offline or upload their own music collection to the cloud. The most popular players in the music streaming market are Spotify (market share of 31% in 2021), Apple Music (market share of 15% in 2021) and YouTube music (market share of 8% in 2021). The global music streaming market accounts for 85% of the music industry revenue. According to a recent KBV research published in January 2021, the global music streaming market size is expected to reach 60,5 billion US dollars by 2026. The rising number of internet/smartphone users, surging digitization and COVID-19 pandemic (during the lockdowns more people found the way to music streaming) are the main reasons for the expansion of the music streaming market.

The music streaming market can be divided into 5 segments: service, content, platform, end user and geography. Based on service type, the market is segmented into on-demand streaming and live streaming. Although the on-demand streaming dominates the global music streaming market, live streaming is expected to have an annual growth rate of 16,9% during 2020-2026. Based on content type, the market is segmented into audio and video and the audio category dominates the market, which is due to the ever-increasing number of commercial users such as cafes, gyms, restaurants, who play songs on their commercial places for entertainment. Based on platform, the market is segmented into apps and browsers and the app category is most used to stream music. Based on end user, the market is segmented into individual and commercial. The commercial end-user category includes places like salons, pubs, restaurants, gyms, and cafes and this category is expected to grow immensely during 2020-2026 due to the raising need of ambiance. Based on geography, the global music streaming market is segmented into North America, Europe, Asia Pacific, and Latin America, Middle East & Africa. North America has the largest revenue share, and it is anticipated that this region will maintain its dominance in 2020-2026. This is due to the presence of big market players such as Apple, Google, Spotify, and Amazon.

The client's challenge: LastFM

LastFM is a well-known music streaming service in Europe. Although recent market research has shown that the market is growing and attracting new customers will contribute to the revenue growth of LastFM, it is paramount as a music streaming service to retain your customers. Therefore, LastFM will have to improve their recommendation systems to provide their customers with a better experience and increase the amount of time spent on their platform.

In this project, we will provide LastFM with a benchmark and create multiple recommendation systems (collaborative-based, content-based and hybrid) utilizing the provided datasets. The project is divided as followed: first we will explain how we reviewed the datasets and created the two base-matrices, then we'll go over the recommendation models which we applied and evaluated. We will end with recommendations related to variety – how LastFM could encourage its users to engage with a variety of artists.

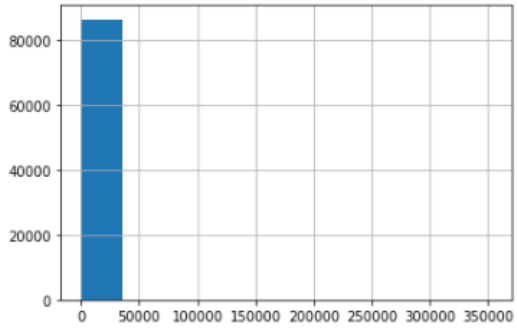
Data preprocessing and creating base matrices

The first step of our project was to read in the four datasets that were provided by LastFM:

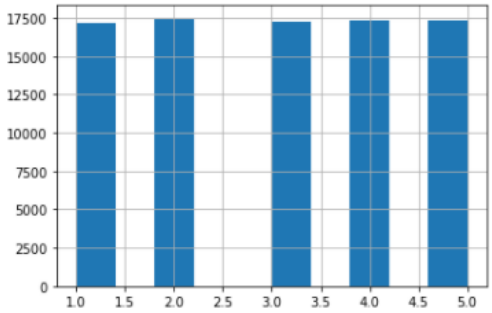
- Artists.dat
- tags.dat
- user_artists.dat
- user_taggedartists.dat

We assume "weight" is a measure of rating and engagement.

We then pre-processed the data and ensured that our two base matrices had the same Items and UserIDs. The pre-processing consisted of a couple of following steps. Firstly, we removed the duplicates of the UserIDs and ArtistIDs. After removing all the duplicates, we had our base matrix for collaborative filtering which we called: colab_filt_df_filtered. Within this base matrix we double checked for duplicates and then after reviewing all the columns, we remarked that the column 'weight' was not nicely distributed, the data was skewed. To resolve this, we categorized the data and added a column 'ratings'. We filtered the values of the 'weight' column into quantiles (1, 0.8, 0.6, 0.4, 0.2) and added a rating to them (for instance 0.8 quantile was rating 4). Now, the values in the 'ratings' column were nicely distributed as seen in the graphs below:



Left: data is skewed in the weight column



Right: data is nicely distributed in the ratings column

After adding the 'ratings' column and making sure the data was nicely distributed, we could delete the 'weight' column.

As a final pre-processing step, we splitted the data of the base-matrix colab_filt_df_filtered in a train dataset (70%) and test dataset (30%). We then imported Reader from the Surprise package to apply this to our train and test dataset.

Recommendation models

Recommendation systems are used to predict the user's habits or to recommend items to users. The number of songs and artists on the music streaming service is rapidly growing, but only a small portion gets the attention, which causes the long-tail effect. Recommendation systems helps to advertise the items with lower popularity. In practice, finding one single ideal model is hard, therefore, to have a better performance its best to combine multiple recommendation models.

We first start with four recommendation systems (User-based, Content-based, matrix factorization and co-clustering) that use collaborative filtering. For these four recommendation systems the base matrix colab_filt_df_filtered was used. Collaborative filtering is most used within recommendation systems because it is applicable in many domains and easy to understand. The basic assumption behind collaborative filtering recommendation systems is that users give rating to catalog items and users who have similar tastes will also have similar tastes in the future. Within our base matrix an interval rating was used (from 1 to 5). We imported Reader from the Surprise package to define the rating scale.

The evaluation of the results of all the models are discussed in a separate section in this report.

User-based

We first applied GridSearchCV and cross validation, to found out the best parameters for User-based:

```
{'k': 30, 'min_k': 10, 'sim_options': {'name': 'pearson', 'user_based': True}}
```

We used these parameters to fit our training dataset and afterwards made predictions on the test dataset. With the User-based recommendation system, we will recommend artists to userX, based on ratings of his/her nearest neighbor. To find similarity in this case we used Pearson correlation, which looks at how much ratings by common users for a pair of items deviate from average ratings for those items. Thanks to the cross validation we knew how many neighbors we should consider (k:30 & min_k:10).

The results of the User-based model are:

	User_based
RMSE	1.402997
MAE	1.195240
Recall	0.021876
Precision	0.673529
F1	0.042376
NDCG@5	0.890630

Item-based

The item-based collaborative filtering recommendation system technique uses the similarity between items (and not users) to make predictions. This is the main difference between the user and item-based techniques, as the implementation is the same.

We also applied GridSearchCV and cross validation, to found out the best parameters for Item-based:

```
{'k': 15, 'min_k': 2, 'sim_options': {'name': 'cosine', 'user_based': False}}
```

We used these parameters to fit our training dataset and afterwards made predictions on the test dataset. To find similarity in this case, we used cosine, which is also known as vector-based similarity, and views two items and their ratings as vectors and afterwards defines the similarity between them as the angle between these vectors. Thanks to cross validation we know how many neighbors we should consider (k:15 and min_k:2).

The results of the Item-based model are:

	Item_based
RMSE	1.018119
MAE	0.778369
Recall	0.488632
Precision	0.867537
F1	0.625153
NDCG@5	0.844591

Matrix factorization

The third recommendation model which we applied is matrix factorization with SVD (Singular Value Decomposition). SVD reduces dimensionality of rating matrices, like our colab_filt_df_filtered matrix. SVD captures important factors/aspects and their weights and assumes that k dimensions capture the signals and filter out noise (K = 20 to 100).

We again used GridSearchCV and CrossValidation to find the best parameters:

```
{'n_factors': 5, 'n_epochs': 20}
```

The results of this third recommendation system are:

	SVD
RMSE	1.245956
MAE	0.909698
Recall	0.353554
Precision	0.897647
F1	0.507299
NDCG@5	0.860893

Co-clustering

For the fourth recommendation system technique, co-clustering, we again applied GridSearchCV and cross validation to find the best parameters to apply:

```
{'n_cltr_u': 10, 'n_cltr_i': 5}
```

We then applied those parameters and gained the following results:

	CoClust
RMSE	1.020859
MAE	0.778142
Recall	0.498089
Precision	0.841646
F1	0.625818
NDCG@5	0.853477

Co-clustering clusters similar users and items based on similarity of their pairwise interactions. This technique allows users or items to be part of multiple clusters at the same time and it can compute the user-item rating estimates for each cluster.

Content-based model

The content-based filtering techniques is mostly used on a specific product page as it recommends products based on similarity between product characteristics. It represents items in an n-dimensional vector space and measures similarity based on cosine. Therefore, the main goal of the content-based recommendation system is to estimate a model to be able to identify similar products.

The content_df_filtered base matrix was used for this technique.

Below are the results for the content-based model:

	Content_based
RMSE	0.901405
MAE	0.683677
Recall	0.502484
Precision	0.915260
F1	0.648782
NDCG@5	0.877049

After applying the 5 recommendation systems mentioned above (4 based on collaborative filtering and 1 based on content), we applied 3 hybrid recommendation systems. A hybrid recommendation system is a special type of recommendation system which can be considered as the combination of the content and collaborative filtering method. Hybrid recommendation systems mitigates the issues related to individual recommendation systems and often improve predictive performance.

Hybrid Model 1: User-based & Item-based

To apply this hybrid recommendation model we first extracted and combined the prediction of content and item-based into a new data frame df_hybrid.

Below are the results of this model:

	Hybrid_UserBased_ItemBased
RMSE	0.958594
MAE	0.760299
Recall	0.102807
Precision	1.000000
F1	0.186447
NDCG@5	0.851058

Hybrid Model 2: Linear Regression

To apply linear regression, we used a train and test dataset.

Below are the results of this model:

	Hybrid_LinearRegression
RMSE	0.900817
MAE	0.687787
Recall	0.000000
Precision	NaN
F1	NaN
NDCG@5	0.858990

Hybrid Model 3: Random Forest

For our final hybrid model, we again used the train and test dataset to apply random forest.

Below are the results of the third hybrid recommendation model, random forest:

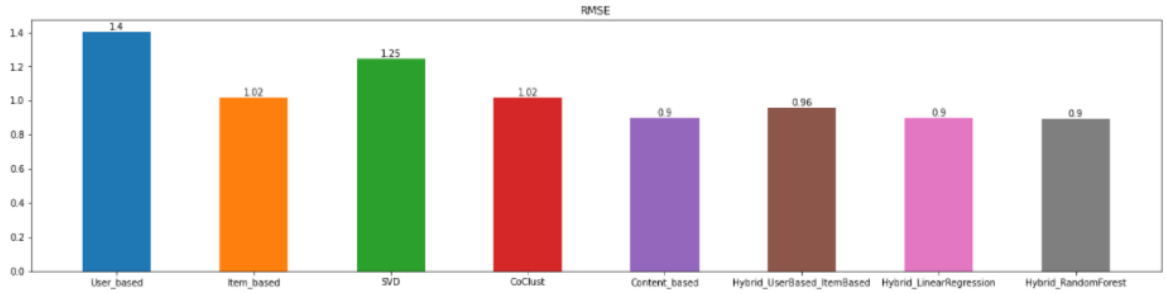
	Hybrid_RandomForest
RMSE	0.895618
MAE	0.687439
Recall	0.000000
Precision	NaN
F1	NaN
NDCG@5	0.878808

Evaluation metrics

Before we'll give an overview of all the evaluation metrics, we would like to dive a bit deeper into each metric separately.

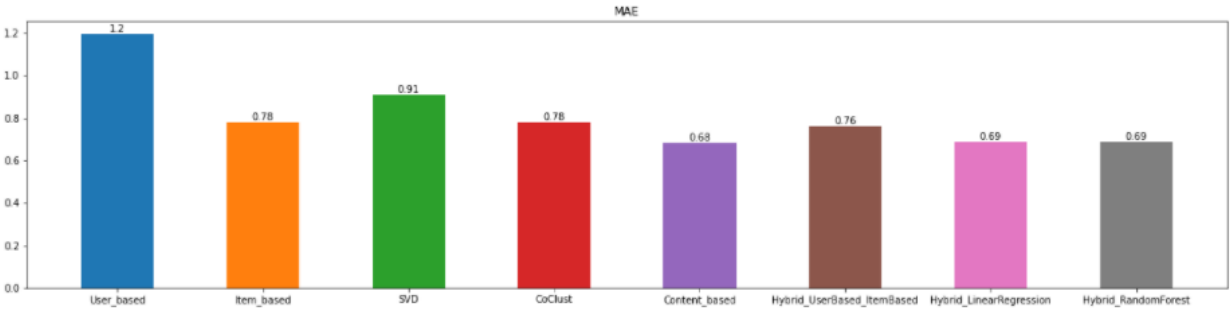
RMSE

RMSE or the Root Mean Squared Error squares the root of mean squared error. The squared error is calculated by taking the differences between the actual and predicted values. In this case, hybrid random forest has the lowest RMSE.



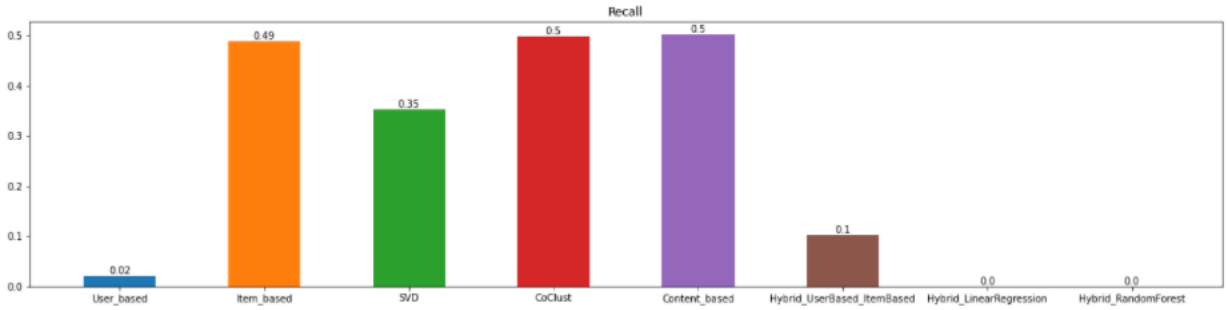
MAE

MAE or the Mean Absolute Error calculates the absolute difference between actual and predicted values and takes the sum f all the errors and divide them by the total number of observations. The main advantage of MAE is that it is very robust to outliers. We aim to get a minimum MAE because this is a loss. In this case, the content-based model has the lowest MAE.



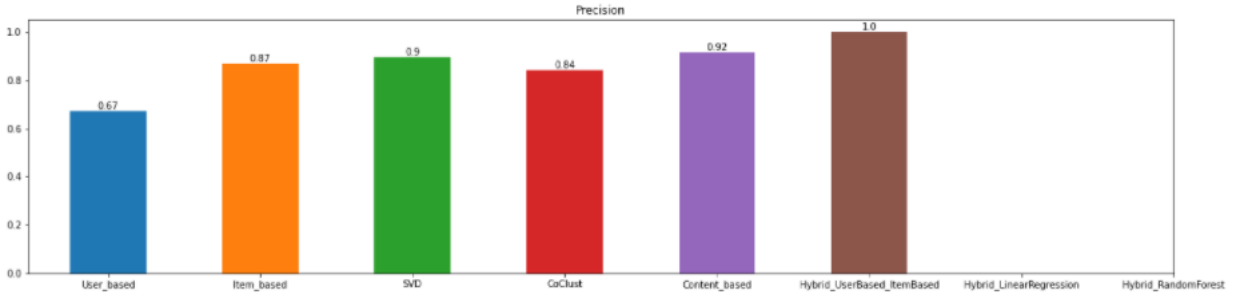
Recall

Recall looks at what proportion of actual positives was identified correctly. Therefore, the higher this score, the better the model is. In this case, the content-based model has the highest recall.



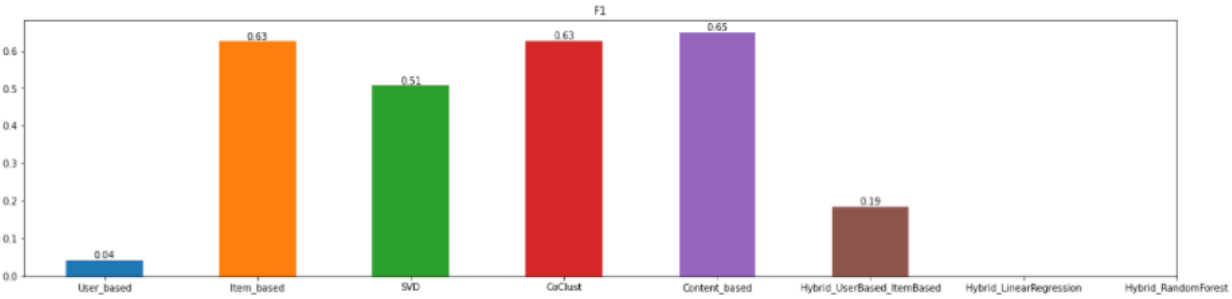
Precision

Precision calculates what proportion of positive identifications was correct. Again, the higher the score, the better the model is performing. In this case, the hybrid model user and content based is the best performing model.



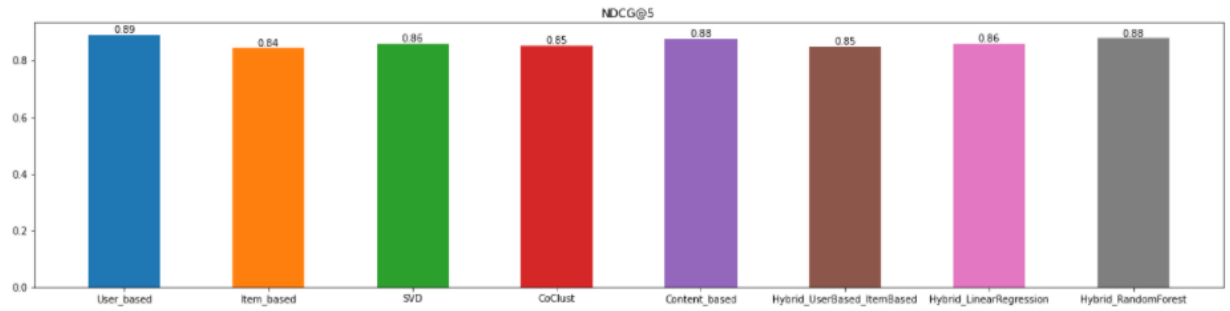
F1

F1 acts a tradeoff between Precision and Recall, mentioned above, and calculates the harmonic mean. Again, the higher this score, the better performing the model is. In this case, content-based has the best F1 score.



NDCG@5

NDCG stands for Normalized Discounted Cumulative Gain. It measures how many of the recommended results are relevant and are showing at the top. The higher the score the better the performance of the model. In this case, the best model is user-based.



Below is a general overview of all the evaluation metrics and scores of all recommendation systems:

	User_based	Item_based	SVD	CoClust	Content_based	Hybrid_UserBased_ItemBased	Hybrid_LinearRegression	Hybrid_RandomForest
RMSE	1.402997	1.018119	1.245956	1.020859	0.901405	0.958594	0.900817	0.895618
MAE	1.195240	0.778369	0.909698	0.778142	0.683677	0.760299	0.687787	0.687439
Recall	0.021876	0.488632	0.353554	0.498089	0.502484	0.102807	0.000000	0.000000
Precision	0.673529	0.867537	0.897647	0.841646	0.915260	1.000000	NaN	NaN
F1	0.042376	0.625153	0.507299	0.625818	0.648782	0.186447	NaN	NaN
NDCG@5	0.890630	0.844591	0.860893	0.853477	0.877049	0.851058	0.858990	0.878808

We can conclude that the content-based collaborative filtering recommendation system is the most accurate one. It has the best scores on 3 out of 6 of the evaluation metrics and is the only model that scores best more than one time on an evaluation metric.

Pros and cons of the models used

Below, we'll give an overview of the pros and cons of all the models we used.

Model	Pros	Cons
Collaborative filtering = User-based, Item-based, Matrix factorization & Co-clustering	❖ Serendipity: the model can help users to discover new interests ❖ No metadate engineering needed ❖ No need for contextual features ❖ Adaptive	❖ Potential danger for the cold-start problem: if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item ❖ Hard to include side features (any features beyond the query or ID) ❖ Long tail problem: skewed distribution
Content-based	❖ No metadata engineering needed ❖ Scalability: no data needed of other users, since the recommendations are specific to this user, therefore easy to scale to many users.	❖ This technique requires a lot of domain knowledge ❖ Potential danger for the cold-start problem: if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item ❖ Overspecialization
Hybrid models: user-based and item-based, linear regression & random forest	❖ Can provide more accurate recommendations ❖ Has the desirable "niche-finding" property which can bring in new items	❖ A lot of domain knowledge is required

Example of how our recommendations would look like for a specific user (user 322)

	user_based	item_based	svd	coclustering
0	Rihanna	Moi dix Mois	Dulce María	DJ Risk One
1	Beyoncé	Lewis Black	Britney Spears	Fyfe Dangerfield
2	Black Eyed Peas	Texas in July	Selena Gomez & the Scene	Eagles of Death Metal
3	Ke$ha	Panda	浜崎あゆみ	陳奕迅
4	Nicki Minaj	The String Quartet	Taylor Swift	Jesse Cook
5	Lily Allen	The Fire Restart	Akira Yamaoka	Ibrahim Ferrer
6	Lady Gaga	Empyrium	Ke$ha	莫文蔚
7	Eminem	平沢進	Miley Cyrus	盧巧音
8	RBD	A Plea for Purging	Christina Aguilera	王菲
9	The Saturdays	Betraying The Martyrs	Katy Perry	Do or Die

From the other users who have similar tastes with that of user 322, we can predict the items that a user might like using a user-based algorithm, and interestingly, the top 10 results tend to be pop artists. Using an item-based algorithm, based on the similarity between items using the ratings that user 322 have given, the top 10 recommendations completely fall out of the pop paradigm, and these recommendations include a wide span of genres of classical, rock, and Japanese animation music. Using SVD, and constructing a matrix with the row of users, columns of items, and the elements of users' ratings, the top 10 recommendation is a combination of pop artists from the user-based algorithm, and Japanese animation music from the item-based algorithm. Finally, exploiting groups of similar users and similar items within calculated clusters, the recommendations have yet a wider range, which includes pop, metal rock, and Chinese pop artists - this user has indeed shown a wide scale of interest!

Recommendations – strategy for encouraging variety and surprise (serendipity)

Recommender Systems are difficult to evaluate when working with offline datasets. The most used metric is root mean squared error, but accuracy is not actually what recommendation systems are looking for, because people don't care about rating you should have given if you have listened to a particular song or artist, instead people care about what the system recommends as the best content for a particular user. In the Netflix prize for recommendation system RMSE was not implemented because it does not matter that much in the real world, what matters is which content is in the list of recommended and how real-world users react to them. As a result of this finding, we will use other metrics too for our models on top of the traditional metrics.

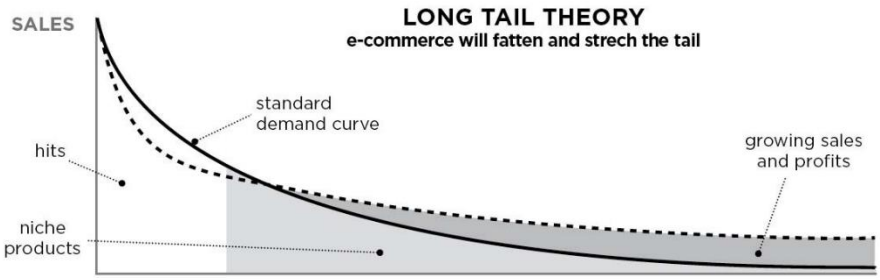
Hit Rate with leave one out cross validation is a better alternative to MAE or RMSE. But it is good to keep in mind that we are not exactly measuring what we want i.e we are predicting on historical data, not on future data (which is possible only through online A/B tests). To measure a Hit Rate, we first generate top N recommendations for all the users in our test data set. If generated top N recommendations contain something that users rated — 1 hit!
With hit rate leave one out cross validation, we compute the top N recommendation list for each user in training data and intentionally remove one of those items form user's training data. We then test our Recommender System's ability to recommend "that" intentionally removed an item in our testing phase. The conclusion is: the greater the Hit Rate, the better our recommender system will be.
In *Cumulative Hit Rate*, we throw away hits of our predicted ratings if they are below some threshold. The main idea is that we should not get credit for recommending items to a user that they won't enjoy. Confined to ratings above a certain threshold. Again, the conclusion is, the higher this score, the better.

The *ARHR or Average Reciprocal Hit Rank* is a variation of hit rate. We now sum up reciprocal of the rank of each hit. It accounts for "where" in the top N lists our hits appear. We get greater credit for successfully recommending item in the top slot than in bottom slot. As it takes rankings into account, the higher the score, the better. Note: it is a more user-focused (in contrast to Hit Rate) metric since users tend to focus more on the beginning of the list.

Coverage is the percentage of possible recommendations (user-item pair) that the system can predict <user, item> pair coverage can be at odds with accuracy. If you enforce a higher quality threshold on recommendations you make, then you might improve your accuracy at the expense of coverage, therefore it is essential to find a good balance. It is a measure of how quickly new items will start to appear in our recommendation list. For example, new books can't enter a recommendation list until someone buys it, and patterns are generated. It measures the ability of the system to recommend long-tail items. The higher the score, the better.

Novelty takes into consideration "How popular are the items you are recommending". Popular items are popular for a reason and hence are good for the recommendation. But there is a challenge to it, namely User Trust. People need to see things they are familiar with to believe those good recommendations are made by the recommendation system. Otherwise, he/she may think that the recommendation system is

bad and will not engage with it or worse, may make fun of it in social media.
Recommendation systems with good novelty scores can make the world a better place. But it is again important to find a good balance between novelty and trust. The lower the score, the better as this tell us we're not recommending long tail items that could hinder trust in the recommender system.
The graph below illustrates *the long tail problem*: The x-axis corresponds to products and y-axis corresponds to the popularity of the product. The shaded region is called "The long tail". Leftmost products in the items are more popular compared to the rightmost items.



Now, let's look at our final overview of the metrics:

	User_based	Item_based	SVD	CoClust	Content_based
RMSE	1.402997	1.018119	0.881286	1.014835	0.901403
MAE	1.195240	0.778369	0.695112	0.773442	0.683674
Recall	0.021876	0.488632	0.459782	0.495606	0.502484
Precision	0.673529	0.867537	0.928971	0.846053	0.915260
F1	0.042376	0.625153	0.615119	0.625060	0.648782
NDCG@5	0.890630	0.844591	0.850307	0.854472	0.876994
Hit Rate	0.007400	0.001057	0.016385	0.003171	NaN
cHR (Cumulative Hit Rate, rating >= 4)	0.016086	0.002681	0.034853	0.008043	NaN
ARHR (Average Reciprocal Hit Rank)	0.004528	0.000125	0.007216	0.001498	NaN
User coverage	0.803383	0.803911	0.494715	0.848309	NaN
Novelty	261.118361	4245.832261	394.785327	6145.931464	NaN

Note 1: We only run these algorithms on our first four models as the hybrid models could not be fitted.
Note 2: When performing sklearn.model_selection.GridSearchCV for the third model in the first part we got parameters that do not optimize RMSE, for this part we corrected this model.

It might seem that the best performing models considering our new metrics are User_based (KNN), having that it has the highest RMSE but the Hit Rate, cHR and ARHR are the second highest and Novelty is the lowest, and SVD, which improves in accuracy metrics but increases Novelty, Hit Rate, CHR and ARHR but decreases in User coverage. If we had to decide, we would select these two models to design an online A/B test. The problem for the SVD approach is that coverage is quite low compared to the other models and our RS would be stuck in one part of the Artist catalog, and if our objective is improving engagement then maybe it would be better to provide recommendations of a more diverse catalog of artists.

Sources

BORCHERT, P. (2022). Recommendation Tools. [Course]. Lille: IESEG Management School. MSc in Big Data Analytics.

O'CALLAGHAN, R., 'Building Recommendation Engines in Python', internet, DataCamp, s.a., (https://app.datacamp.com/learn/courses/building-recommendation-engines-in-python)

ANONYMOUS, 'music streaming service', internet, PC Mag Encyclopedia, s.a., (https://www.pcmag.com/encyclopedia/term/music-streaming-service)

ANONYMOUS, 'Global music streaming market report, internet, KBV Research, 2021-01-02, (https://www.kbvresearch.com/music-streaming-market/)

MLADENOV, P., 'These are the most used music streaming services in the world', internet, Phone arena, 2022-01-20, (https://www.phonearena.com/news/these-are-the-most-used-music-streaming-services-in-the-world_id137933)

DAIKAWA, J., 'Building (and Evaluating) a Recommender System for Implicit Feedback', internet, Medium, 2020-02-01, (https://medium.com/@judaikawa/building-and-evaluating-a-recommender-system-for-implicit-feedback-59495d2077d4)

B, A., 'Recommender Systems — It's Not All About the Accuracy', internet, Gab 41, 2016-01-27, (https://gab41.lab41.org/recommender-systems-its-not-all-about-the-accuracy-562c7dceeaff)

ANONYMOUS, 'Content-based Filtering Advantages & Disadvantages', internet, Developers Google, 2018-11-30, (https://developers.google.com/machine-learning/recommendation/content-based/summary)

KANE, F., 'Building Recommender Systems with Machine Learning and AI', internet, Sundog Education, 2021-04-13, (https://sundog-education.com/course/building-recommender-systems-with-machine-learning-and-ai/)

BROWNLEE, J., 'Tune Hyperparameters for Classification Machine Learning Algorithms', internet, Machine Learning Mastery, 2019-12-13, (https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/)

ANONYMOUS, 'Evaluating Recommendation Systems — Part 2', internet, Medium, 2019-06-01, (https://medium.com/fnplus/evaluating-recommender-systems-with-python-code-ae0c370c90be)