COLOMBIER Rémi

GHOSLYA Aazad

SRIPADA Sumanth

Professor: Philipp BORCHERT

Group assignment recommendation System

LastFM case study

X

TABLE OF CONTENTS

INTRODUCTION

We live today in a society where recommendation algorithms are omnipresent in our daily lives. And this, in all sectors, sports, retail, e-commerce, video streaming, music, social networks, the press.

In this study we will analyze different recommendation algorithms to improve the current recommendation system that recommends the 10 most popular artists to all users.

Our study aims to improve the user experience and continuously increase performance (e.g. listening time, shortening of music search times, variety of music listened to, etc.).

Our recommendation tools will also make it possible to manage a large volume of data that is impossible to process manually and to make relevant recommendations to users based on their interests. Throughout the study, we will propose different algorithms because they process results that are more relevant than others, depending on the situation and the company's challenges.

For example, if we take the example of a user-based algorithm, the negative point is that this user will not get any music recommendations because he will not have listened to any sound on the platform yet.

Data

The entire data is consisting of 4 datasets.

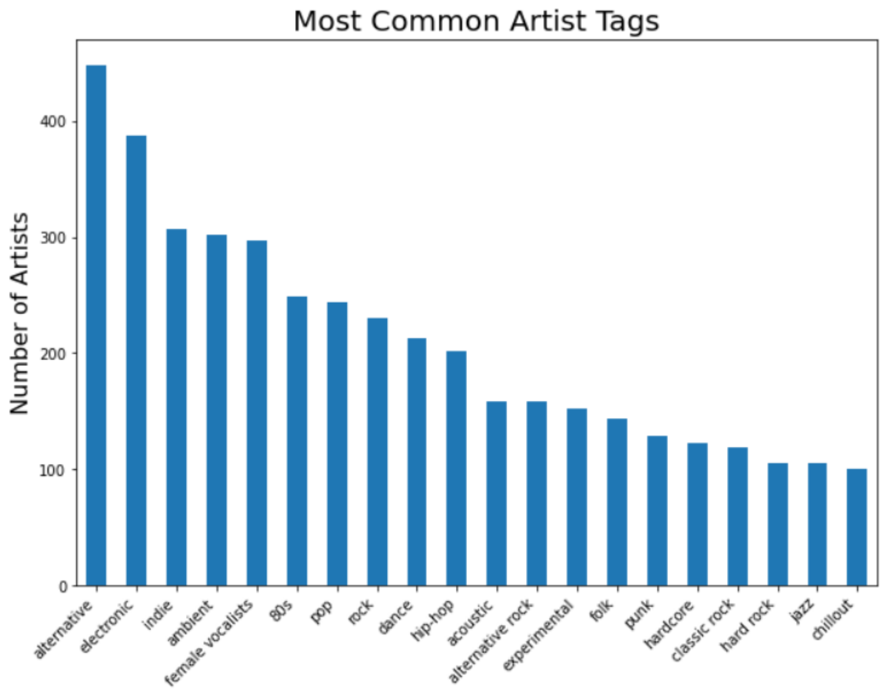
1. Artists: The dataset mainly consists of ArtistID, Artist Name, Url and pictureURL. There is total 17632 unique artists.
2. User_Artists: This dataset consists of userID, artistID, weights. The weight corresponds to the number of times particular user listens to the corresponding artist.
3. User_Tags : This dataset consists of userID, artistID, tagID, day, month, year. This dataset consists of the information of particluar user there corresponding artist and tag along with day and month and year the user listens to the artist with particular tag.
4. Tags: This dataset consists of tagID and tagValue column. There are in total 11946 tagID’s.

Data Visualization:

To attain the proper insights of the data and get the more insights regarding the artists the relationship between user's artist's and tags we have plotted few graphs.

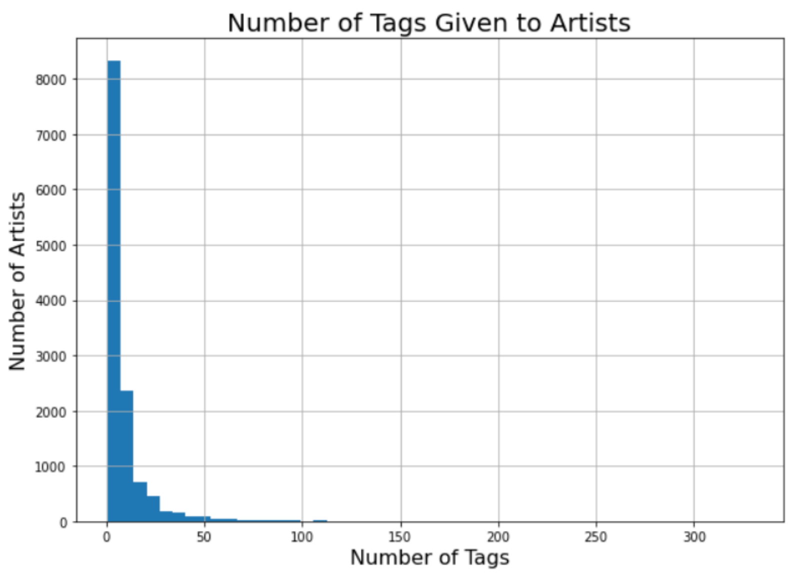
Plot I:

Most Common Artists Tags: This is the representation of tags and the number of artists



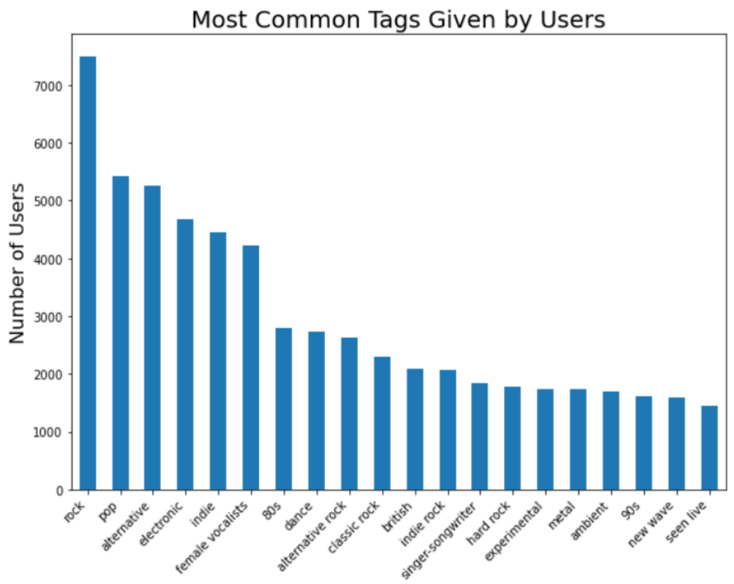
Plot II:

This plot shows the relation between Number of tags and Number of Artists.



PLOT III:

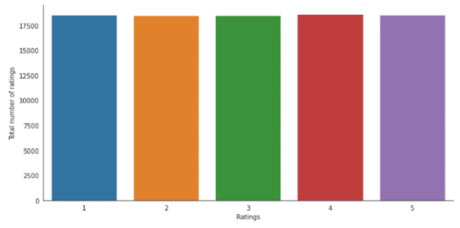
This plot shows the relation between Number of Users and tags. This is used for getting the visualization of Most common used tags by users.



A) Preprocessing steps

- To conduct our analyses with the different algorithms it is interesting to have a clean database with relevant variables. For this we merged different datasets together and standardized everything to the standards of the algorithms so that there are no errors when we run the recommendation algorithms. We have selected only the variables that are interesting for our UserID, artistID and rating algorithms.

- We transformed the "weight" variable so that it took the form of a rating variable from 1 to 5. Thus, we took the maximum and minimum of weight and divided it into quantiles to have it replaced by the 1-5 rating variable. Basically, min and max scaling is to handle the skewness in the weights.



- This graph shows us the distribution of the ratings variable. It is this rating that will allow us to make targeted recommendations. Then we split our dataset into 80% train and 20% test. Then we created the Reader which will allow us to evaluate the scores of each user.

B) Collaborative filtering :

The first type of algorithm we will use is collaborative filtering. This algorithm is the most efficient when it is necessary to make recommendations to a client whose preferences and interests are known, when we have data on a client. The recommendation algorithms for collaborative filtering are diverse and can be based on the personality of the users (user-based), on the characteristics of the music (item-based) or even by doing matrix factorizations (SVD).

a) User-Based :

Working :

- This first technique is aptly named as it recommends music based on the common preferences of users, their profiles.
- The principle is simple: if two users both like music by David Guetta and another by Coldplay, then they have a similar profile. And if it turns out that one of these users likes Daft Punk music that the second user has not yet listened to, then our algorithm will recommend the Daft Punk music to the second user because they have previously had identical preferences.
- The figure on the right show perfectly this mechanism and the algorithm used here is the KNN algorithm of the "Surprise" package. In the KNNBasic parameters you just need to set the ***'****user_based* = *True'* so that the algorithm understands that it is user-based.

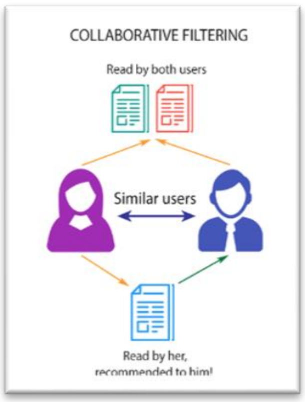
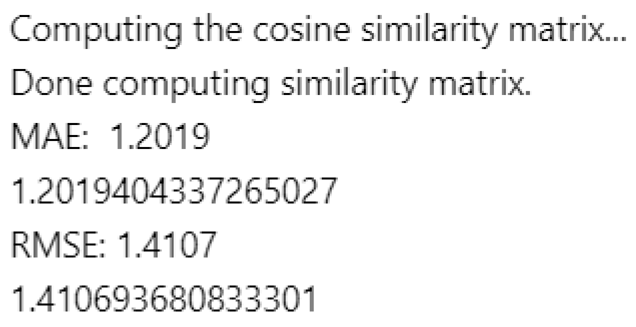


Figure 1: Users-based Collaborative filtering

Output:

We got a the following RMSE and MAE



```
Computing the cosine similarity matrix...
Done computing similarity matrix.
MAE:  1.2019
1.2019404337265027
RMSE: 1.4107
1.410693680833301
```

b) Item-Based

Working :

- Here the algorithm will work like user-based but will now use the similarities between the characteristics of the music. For example, let's suppose that John likes rock and classical music, Louis likes rock, rap and reggae, Martin likes rap and classical music, and Antoine likes only reggae.

- To recommend a music to Martin, we will compare the similarities with the other items not watched by Martin.

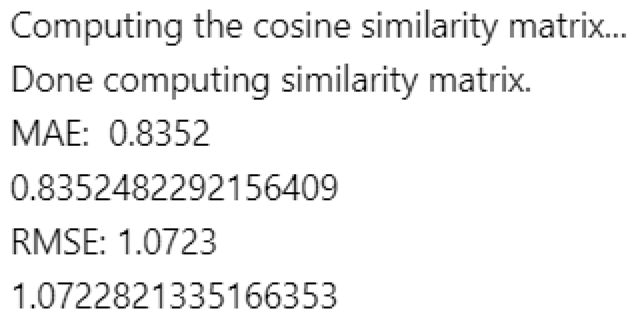
Rock	Jean, Louis
Classic	Jean, Martin
Rap	Louis, Martin
Reggae	Louis, Antoine

- Martin has been watching classical and rap music and we're wondering whether to recommend rock or reggae, we're calculating similarity scores for rock and reggae.
    - *score(rock) = similarity (rap, rock) + similarity (Classique, rock)*
    - *score(reggae) = similarity (rap, reggae) + similarity (Classique, reggae)*

- Then we recommend the item with the highest score to the user.
- These two algorithms work in the same way but do not have the same starting point. One will use user profiles to make recommendations while the other will use item profiles (music in our case). Both algorithms are used depending on the usage situation.
- For example, if a customer has not yet viewed much music, then we will use an item-based algo to recommend music rather than a user-based one. In the KNNBasic parameters we just need to set the argument '*user_based = False*' so that the algorithm understands that it is item-based.

Output:

We got a the following RMSE and MAE.



```
Computing the cosine similarity matrix...
Done computing similarity matrix.
MAE:  0.8352
0.8352482292156409
RMSE: 1.0723
1.0722821335166353
```

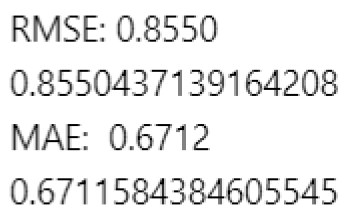
c) Singular Value Decomposition

Working :

- The Singular Value Decomposition is a method used as a dimension reduction technique. SVD will reduce the number of dimensions from N dimension to K dimension with K < N.
- This algorithm is part of the collaborative filtering algorithms. In fact, the main idea is to transform models that take a long time to run into fast models with few dimensions and therefore faster predictions.
- The creation of a dimension will capture important information and will evaluate the weight of this information for the user.
- In fact, this dimension reduction should be seen as new variables that will capture the maximum amount of information and filter out noisy variables. These algorithms are also very efficient to obtain a score for each item and each user.
- In fact, we must see a matrix with in the first column the items (music), and in the second column the users. The values correspond to the ratings. Then there is a line for each combination of item-users, so we will end up with blank boxes because obviously not all users have listened to all the music. And this is where the algorithm comes in because it will fill in these blank spaces in this matrix.
- The algorithm can use different methods such as ALS or SVD which decompose the matrices into several small matrices.

Output:

We got a the following RMSE and MAE



```
RMSE: 0.8550
0.8550437139164208
MAE:  0.6712
0.6711584384605545
```

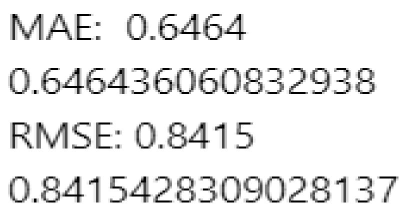
d) SVDpp

Working :

- It is simply an extension of the SVD algorithm that considers implicit ratings. SVD++ achieves better recommendation accuracy by adding implicit feedback information like, artists that the user has evaluated, and the specific value of the score does not matter for this kind of information.
- That is, it adds a factor vector for each item, and these item factors are used to describe the characteristics of the item, regardless of whether it has been evaluated.

Output:

We got a the following RMSE and MAE



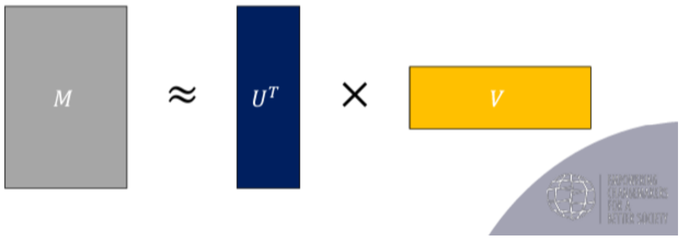
```
MAE:  0.6464
0.646436060832938
RMSE: 0.8415
0.8415428309028137
```

e) ALS

Working

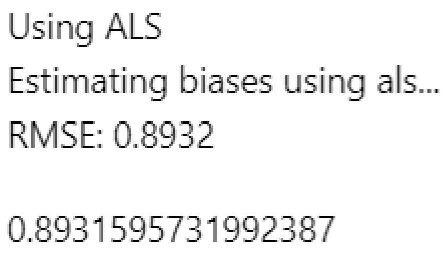
- The ALS (Alternating Least Squares) algorithm works by simply optimizing the ratings obtained either according to users or according to items.
- The objective of this algorithm is to represent several small matrices through ALS with users and products so that this matrix is representative of the original database.
- ALS will decompose our matrix of ratings M into two other matrices. A matrix U which will correspond to the user's matrix. And a matrix V which will correspond to the item's matrix.
- This will minimize the average of the errors in the square of the observed scores and therefore obtain more accurate predictions. The diagram below illustrates the decomposition of the matrix.

$$M \approx U^T V$$



Output:

We got a the following RMSE and MAE

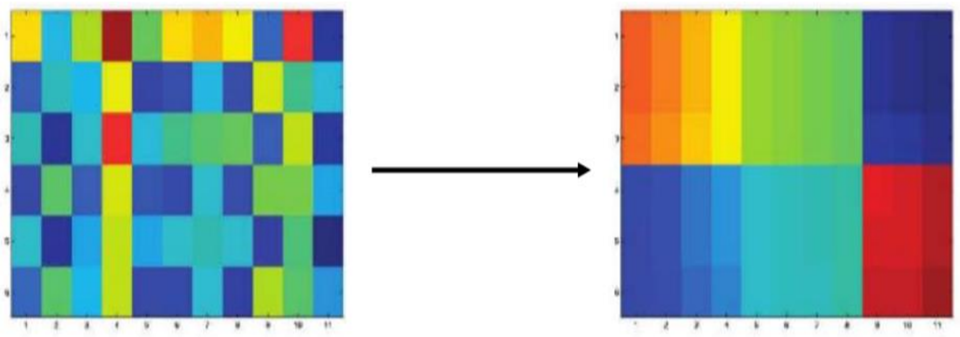


```
Using ALS
Estimating biases using als...
RMSE: 0.8932

0.8931595731992387
```

f) Co clustering

Working

- In the world of data, clustering means to group individuals/items according to their similarities so that objects that look alike belong to the same group.
- But today when we look at the notes that a user can put on music, it can either be according to the user's profile and preferences, or according to the music itself and its characteristics. The algorithm co clustering creates clusters across the users-item matrix.
- The idea is to make clusters according to the similarities of the pairwise interactions. This algorithm allows users or items to be part of several clusters at the same time, hence the name co-clustering. Then we calculate the estimated ranking of each user or item for each cluster.
- The graph below explains the mechanism of co-clustering, with on the left before co clustering and on the right with the co cluster done.



Output:

We got a the following RMSE and MAE

```
RMSE: 1.0216
1.0216057147644062
MAE:  0.7862
0.7862178637979814
```

C) Content-based

Working

- This is another recommendation technique that relies on content and features to recommend content that you will most likely like.
- In fact, the goal is to classify products with keywords, then identify what the user likes as keywords and then recommend music with the keywords they like. The figure on the right illustrates well the mechanism of the content-based filtering approach.
- The recommendations are made by inspecting the content of a music and comparing all the content of other music. So, we will count the number of similar keywords per content between the different music of the platform to be able to recommend to the users some music they haven't used yet, and which have a similar content to what they already like as music.
- To do this we will tokenize the '*tagValue*' column, put them in lowercase, remove the stop words, do stemming, etc.... All of this allows us to have a more efficient and accurate model.

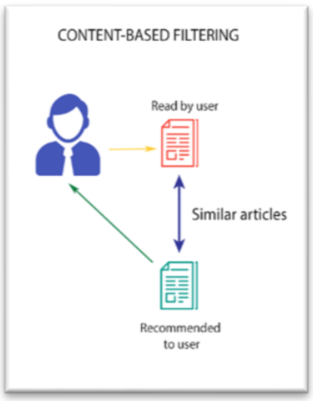
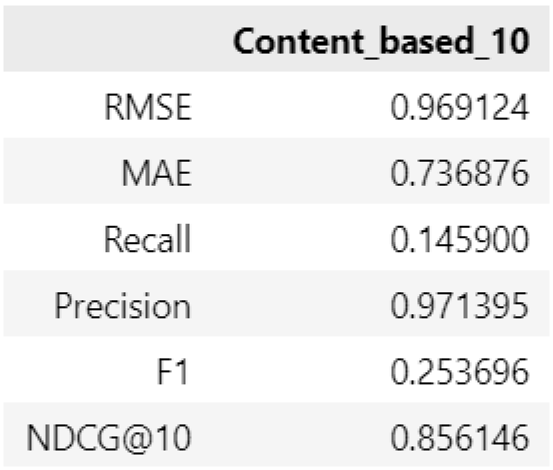


Figure 2: Content-based filtering

Output:

We got a the following RMSE and MAE

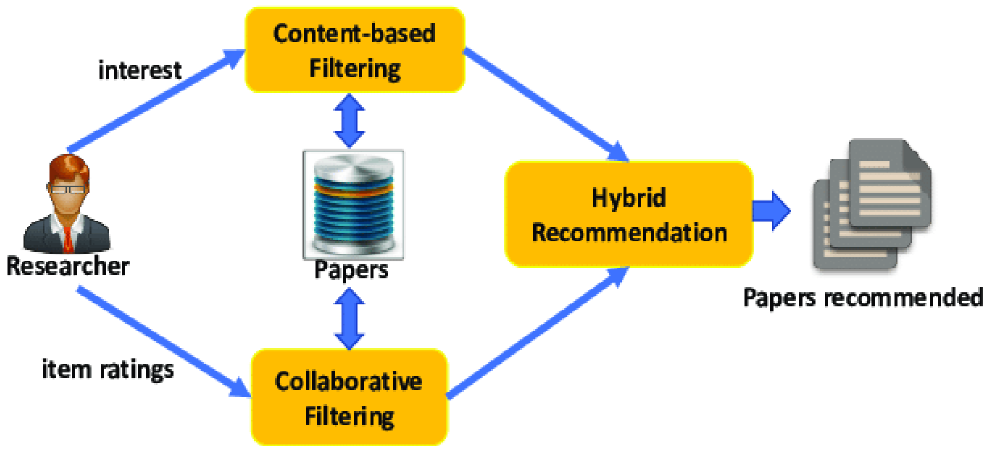


	Content_based_10
RMSE	0.969124
MAE	0.736876
Recall	0.145900
Precision	0.971395
F1	0.253696
NDCG@10	0.856146

D) Hybrid

Working

- Hybrid Recommendation System is another approach which is the combination of content based and Collaborative filtering method.
- Combining both the approaches often improves the predictive performance of the model and mitigates the issues related to individual recommendation systems. Like in case of content based it is hard to recommend for new users and even the collection of information is hard. Its similar for the collaborative filtering method too in case of new users and its bit unstable compared to other approaches.
- Here we can implement the Hybrid approach by using content and collaborative methods to generate predictions separately and then combine them as features to the Hybrid model.

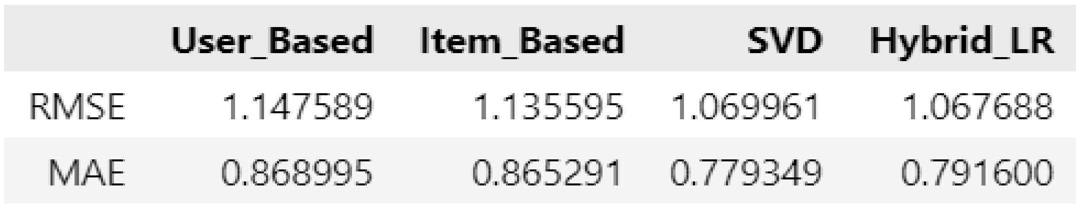


Source : Analyticsmarg.com

Hybrid Model -1 :

Here in this case, we took the predictions from the Item Based and Content Based Approaches and initially selected the Linear Regression Model to predict the Recommendations.

- We have noticed that the Hybrid Linear model performed better than the Collaborative Filtering Algorithms:

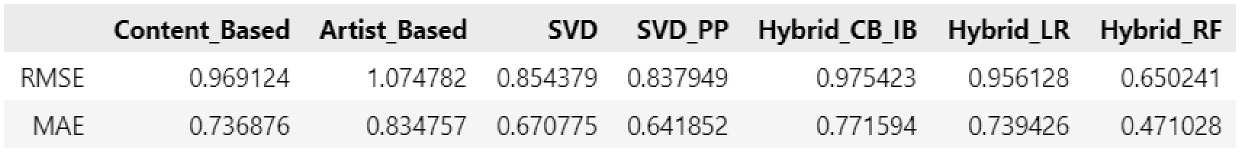


	User_Based	Item_Based	SVD	Hybrid_LR
RMSE	1.147589	1.135595	1.069961	1.067688
MAE	0.868995	0.865291	0.779349	0.791600

Hybrid Model – 2 :

Similarly, we have tried the **Random Forest** Model with the predictions from Item Based and Content Based Approaches to predict the recommendations.

- Random Forest is the best performing model so far among all the models we have tried with the RMSE 0.65.



	Content_Based	Artist_Based	SVD	SVD_PP	Hybrid_CB_IB	Hybrid_LR	Hybrid_RF
RMSE	0.969124	1.074782	0.854379	0.837949	0.975423	0.956128	0.650241
MAE	0.736876	0.834757	0.670775	0.641852	0.771594	0.739426	0.471028

E) Benchmark of the model

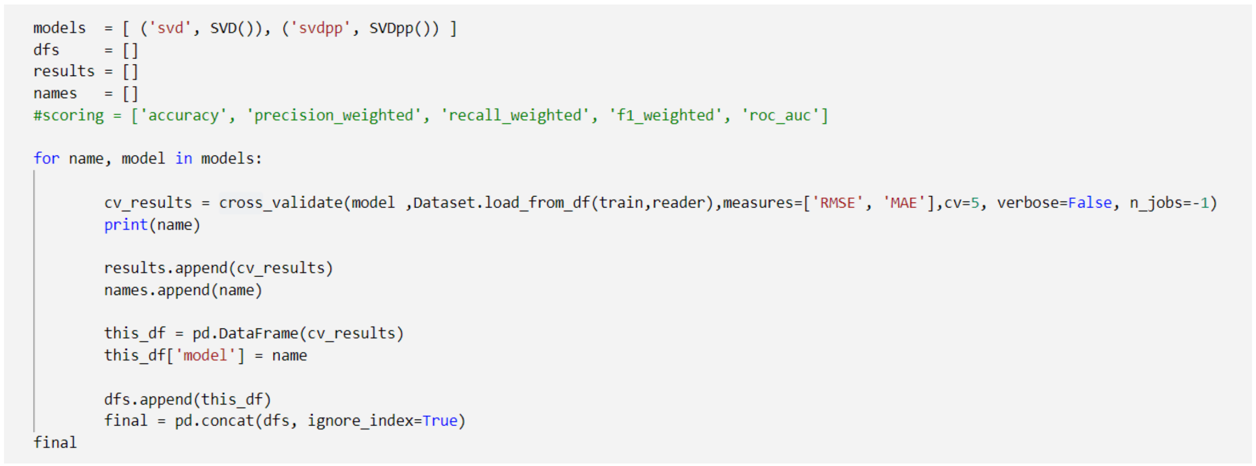
Algorithms		Advantages	Disadvantages
Non-personnalized	Non-personnalized	Recommend popular items, best ratings, most frequently used	Not relevant for the users He can have some recommendation of some music he will hate
Collaborative filtering	KNN - User-based	User will discover new products. Interesting for the firm since it can show new products to their clients.	Need of history to recommend No suitable for new client
	KNN - Item-based		Music that hasn't been listen yet can't be recommended.
	Co clustering	Simultaneous clustering of similar pairwise interactions.	It is a costly technique since we need excellent hardware and more servers.
	ALS	Efficient with large volume of data, since it reduces the number of dimensions Dimension reductions make the algorithm faster to compute.	Not as accurate as the other models.
	SVD	Extremely useful when the matrix is sparse a lot. Perform well in decomposing the matrix to find dimension with le largest variance.	SVD is computationally expensive and slow. Need to be careful with missing data.
	SVDpp	Same as SVD and consider implicit ratings	More time consuming to compute
Content based	Content-based	New items can be recommended Satisfaction of the client for what he sees since he like it	Time consuming in preprocessing Users won't have any recommendation for different items (variety).

Cross Validation :

Working :

CROSS VALIDATION SVD

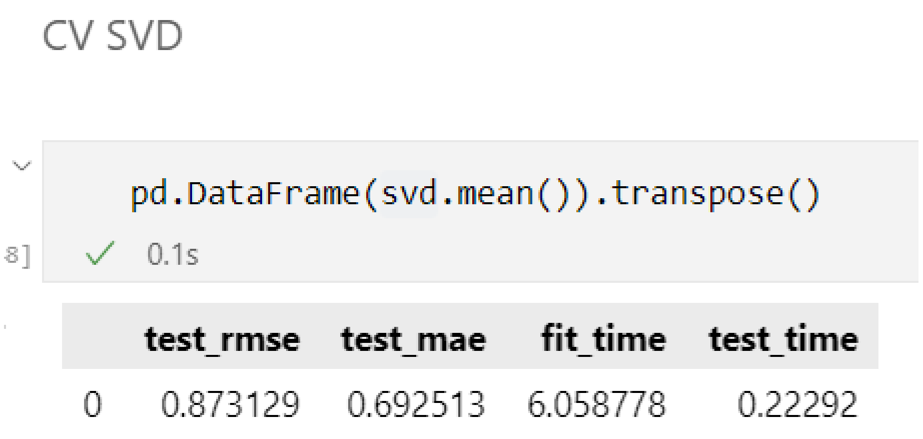
- Cross-validation, sometimes called rotation estimation or out-of-sample testing, is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set.
- From the results it is observed that the SVD and SVDpp are the best performing Models and below is the cross Validation for both the models.



```
models  = [ ('svd', SVD()), ('svdpp', SVDpp()) ]
dfs     = []
results = []
names   = []
#scoring = ['accuracy', 'precision_weighted', 'recall_weighted', 'f1_weighted', 'roc_auc']

for name, model in models:

        cv_results = cross_validate(model ,Dataset.load_from_df(train,reader),measures=['RMSE', 'MAE'],cv=5, verbose=False, n_jobs=-1)
        print(name)

        results.append(cv_results)
        names.append(name)

        this_df = pd.DataFrame(cv_results)
        this_df['model'] = name

        dfs.append(this_df)
        final = pd.concat(dfs, ignore_index=True)
final
```

Output

- ➢ **CROSS VALIDATION SVD**



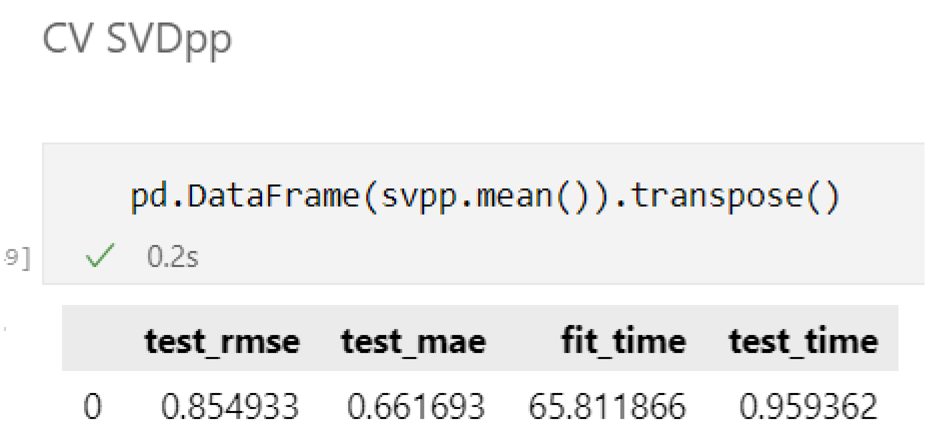
CV SVD

```
pd.DataFrame(svd.mean()).transpose()
```

8] ✓ 0.1s

	test_rmse	test_mae	fit_time	test_time
0	0.873129	0.692513	6.058778	0.22292

➢ **CROSS VALIDATION SVDpp:**



CV SVDpp

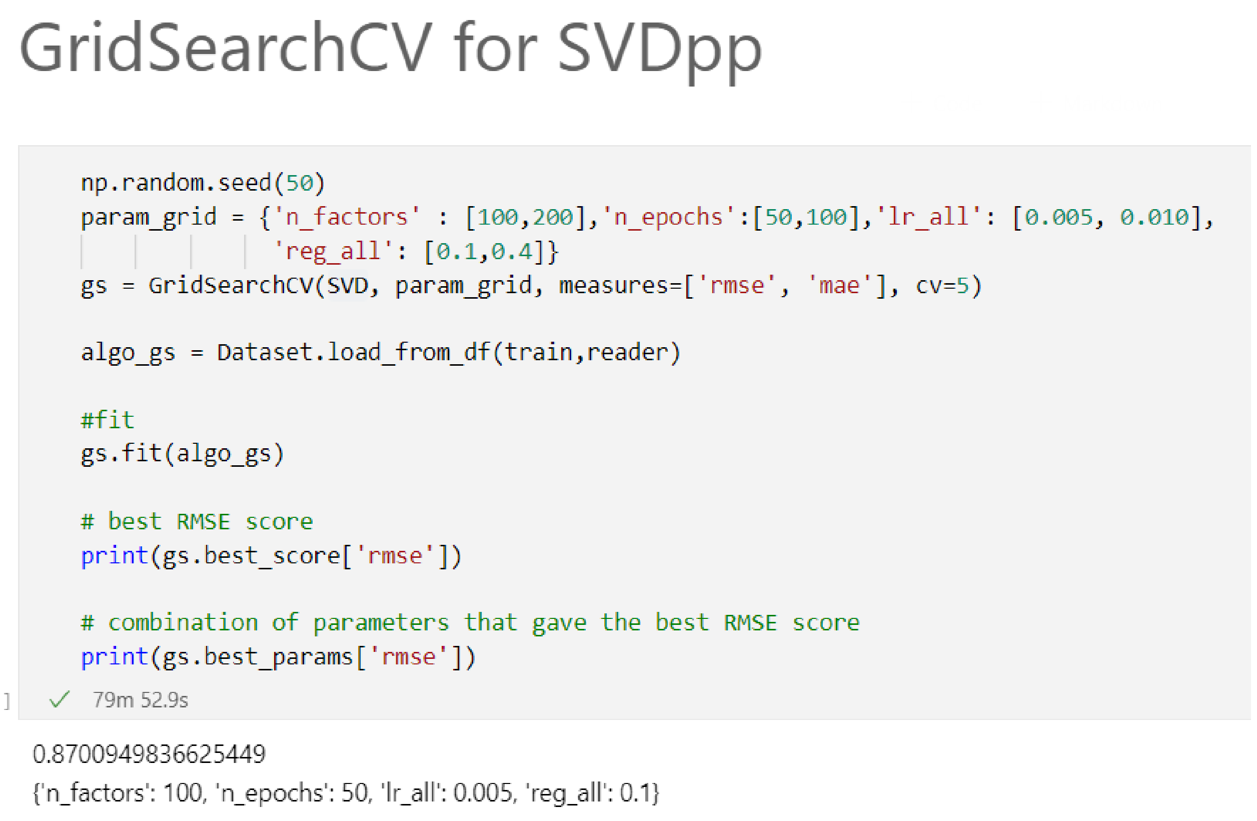
```
pd.DataFrame(svpp.mean()).transpose()
```

9] ✓ 0.2s

	test_rmse	test_mae	fit_time	test_time
0	0.854933	0.661693	65.811866	0.959362

Grid Search CV

It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set.



GridSearchCV for SVDpp

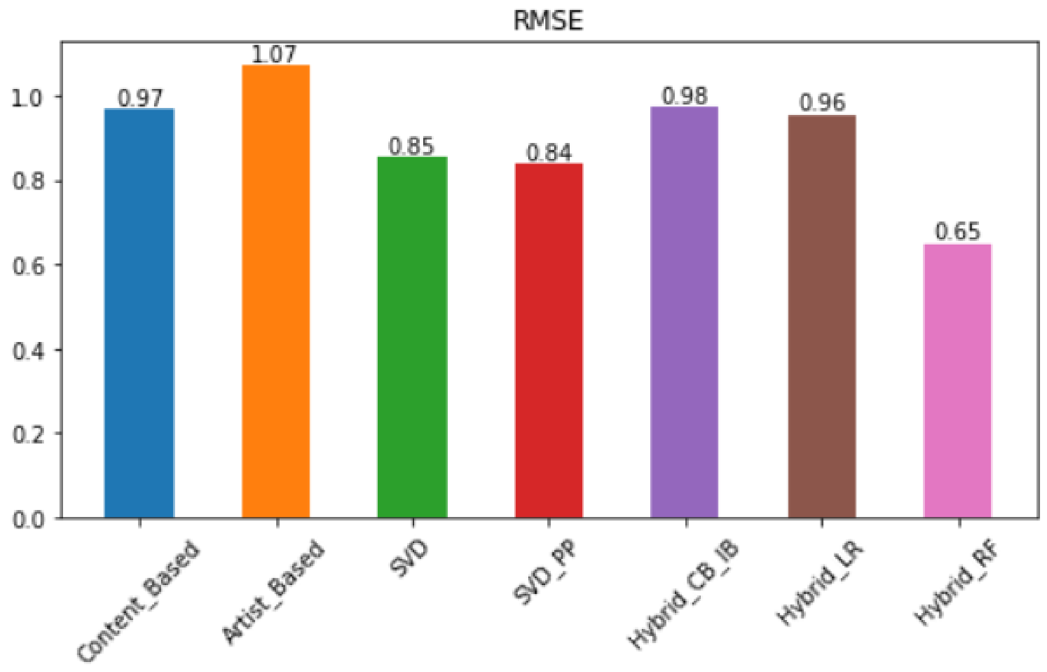
```
np.random.seed(50)
param_grid = {'n_factors' : [100,200],'n_epochs':[50,100],'lr_all': [0.005, 0.010],
              'reg_all': [0.1,0.4]}
gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=5)

algo_gs = Dataset.load_from_df(train,reader)

#fit
gs.fit(algo_gs)

# best RMSE score
print(gs.best_score['rmse'])

# combination of parameters that gave the best RMSE score
print(gs.best_params['rmse'])
```

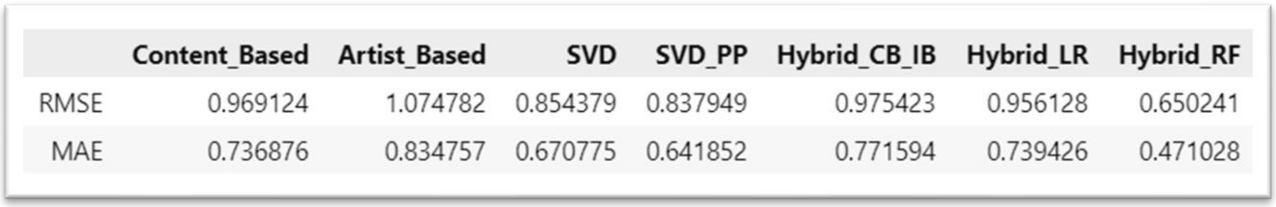
```
0.8700949836625449
{'n_factors': 100, 'n_epochs': 50, 'lr_all': 0.005, 'reg_all': 0.1}
```

F) Evaluation of Our Models :

Below is a table with the scores obtained for the different recommendation algorithms. We can observe that the lowest RMSE is obtained with the Hybrid Random Forest Model after the parameter tuning with the RMSE of 0.65.

- The second-best performing Model is the SVDpp with the RMSE of 0.84.
- It is therefore with this algorithm that we will carry out a qualitative study on the recommendations to a user.





	Content_Based	Artist_Based	SVD	SVD_PP	Hybrid_CB_IB	Hybrid_LR	Hybrid_RF
RMSE	0.969124	1.074782	0.854379	0.837949	0.975423	0.956128	0.650241
MAE	0.736876	0.834757	0.670775	0.641852	0.771594	0.739426	0.471028

G) Qualitative assessment:

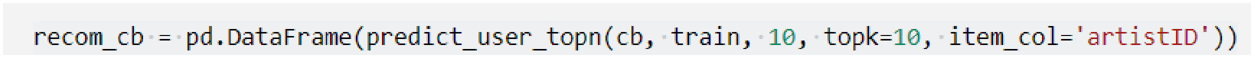
Recommendation of Artists for a Specific User :

Here, we have tried to recommend the Artists for one particular user based on the results we get and tried to see what the recommendations are and what actually the user listens to and rated Highest among the artists that he prefers and see if there is any similarity or pattern in the recommendations.

Function – **predict_user_topn** is used to recommend the artists for users based on the model.

Within this function we can select the

- Model with which we want to recommend the Artists
- the Data on which we have fit the Model
- UserID
- Top_n number , 10 in our case
- Item_col , which is artistID in our case



```
recom_cb = pd.DataFrame(predict_user_topn(cb, train, 10, topk=10, item_col='artistID'))
```

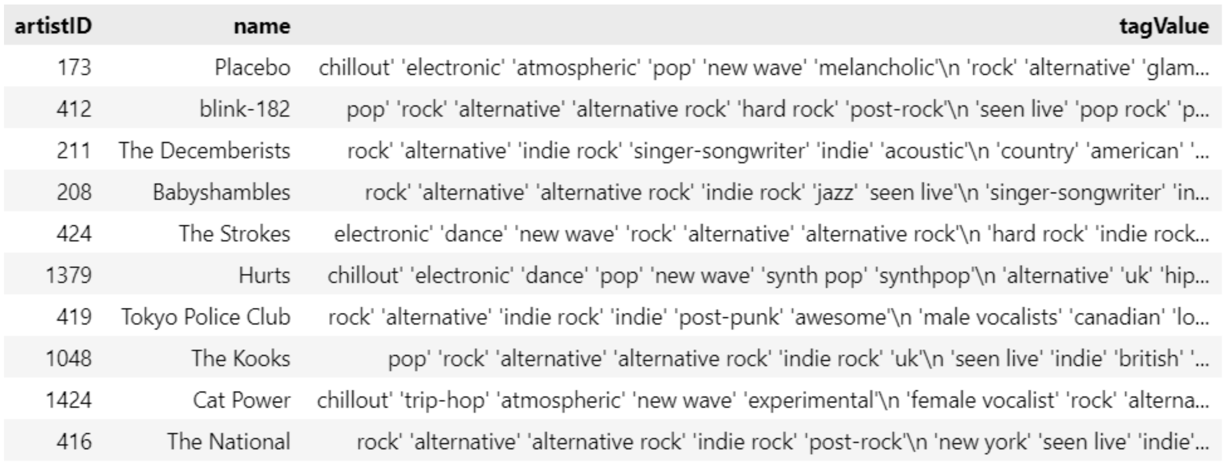
This recommends top 10 artists for the random user 10, and we got the name and tagValue of the Recommended artists by merging the recommended artists with the artists and tags data.

Steps Followed to get this:

- First Using the Function **predict_user_topn** got the 10 recommended artists for one selected User.
- Merged the recommended artists dataframe with the artists data to get the name of the recommended Artist.
- Merged this again with the tags_grouped which has the tagValues per artists.

This recommends top 10 artists for the random user 10 , and we got the name and tagValue of the Recommended artists by merging the recommended artists with the artists and tags data.

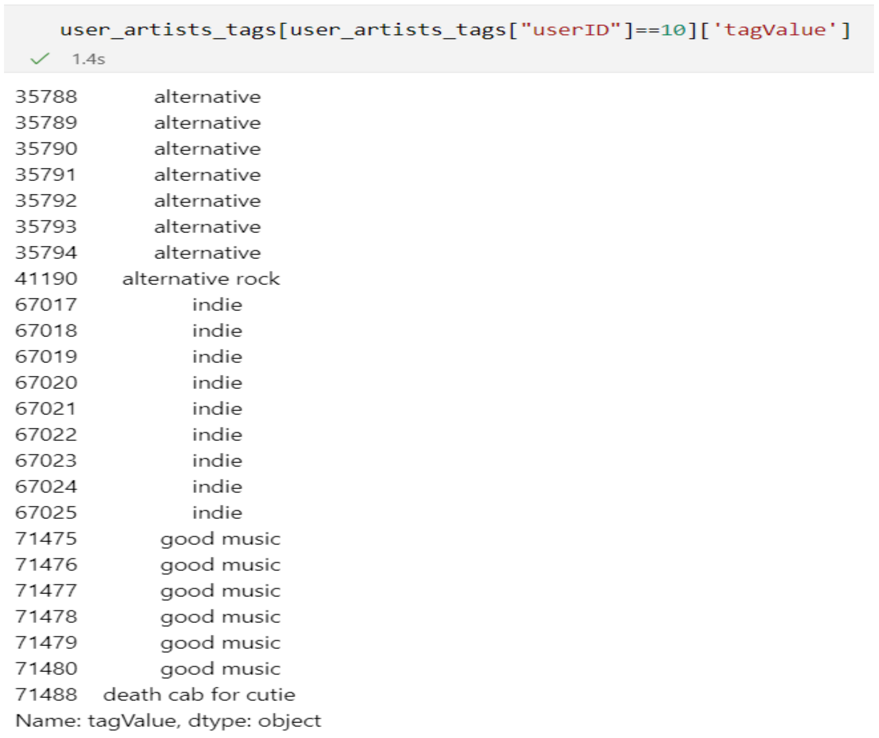
Results :



artistID	name	tagValue
173	Placebo	chillout' 'electronic' 'atmospheric' 'pop' 'new wave' 'melancholic'\n 'rock' 'alternative' 'glam...
412	blink-182	pop' 'rock' 'alternative' 'alternative rock' 'hard rock' 'post-rock'\n 'seen live' 'pop rock' 'p...
211	The Decemberists	rock' 'alternative' 'indie rock' 'singer-songwriter' 'indie' 'acoustic'\n 'country' 'american' '...
208	Babyshambles	rock' 'alternative' 'alternative rock' 'indie rock' 'jazz' 'seen live'\n 'singer-songwriter' 'in...
424	The Strokes	electronic' 'dance' 'new wave' 'rock' 'alternative' 'alternative rock'\n 'hard rock' 'indie rock...
1379	Hurts	chillout' 'electronic' 'dance' 'pop' 'new wave' 'synth pop' 'synthpop'\n 'alternative' 'uk' 'hip...
419	Tokyo Police Club	rock' 'alternative' 'indie rock' 'indie' 'post-punk' 'awesome'\n 'male vocalists' 'canadian' 'lo...
1048	The Kooks	pop' 'rock' 'alternative' 'alternative rock' 'indie rock' 'uk'\n 'seen live' 'indie' 'british' '...
1424	Cat Power	chillout' 'trip-hop' 'atmospheric' 'new wave' 'experimental'\n 'female vocalist' 'rock' 'alterna...
416	The National	rock' 'alternative' 'alternative rock' 'indie rock' 'post-rock'\n 'new york' 'seen live' 'indie'...

Here we can see the artistID, name and tagValue(s) of the recommended Artists by our Model. We have compared this with the actual artist this particular user have already rated and we have noticed that there is a match in the tagValues. This suggests our model recommendations have similarities with the likes of the User.

Ex: Artists tagsValue that the user 10 has already rated:



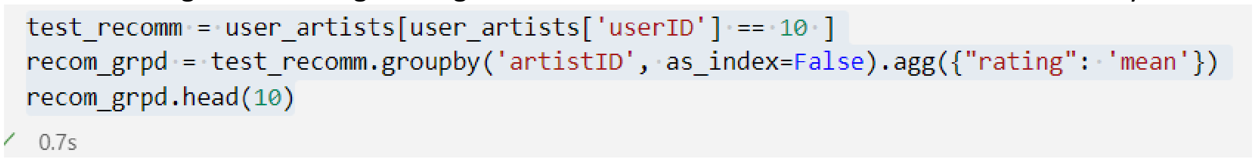
```
user_artists_tags[user_artists_tags["userID"]==10]['tagValue']
✓ 1.4s
```

```
35788          alternative
35789          alternative
35790          alternative
35791          alternative
35792          alternative
35793          alternative
35794          alternative
41190      alternative rock
67017               indie
67018               indie
67019               indie
67020               indie
67021               indie
67022               indie
67023               indie
67024               indie
67025               indie
71475          good music
71476          good music
71477          good music
71478          good music
71479          good music
71480          good music
71488    death cab for cutie
Name: tagValue, dtype: object
```

➢ Recommendation Strategy for the Variety of the Recommendations by our Model :

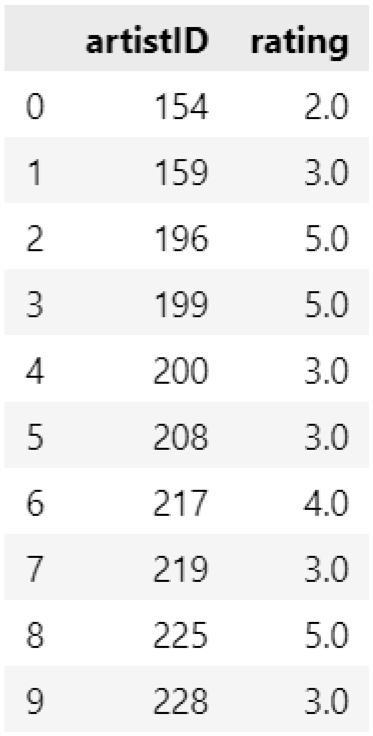
Here we are taking one userID 10 and aggregating all the ratings(mean) of that user for each of these artists which are actually recommended by our Model. So, the aim is having the Variety of suggestions instead of Recommending the already rated artists by the user.

From this we got the average rating of all the artists that the user 10 have rated already.



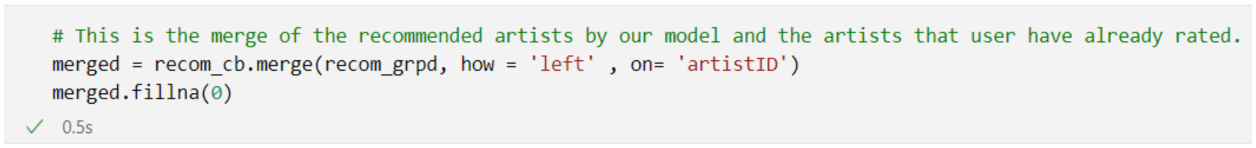
```
test_recomm = user_artists[user_artists['userID'] == 10 ]
recom_grpd = test_recomm.groupby('artistID', as_index=False).agg({"rating": 'mean'})
recom_grpd.head(10)
```

✓ 0.7s

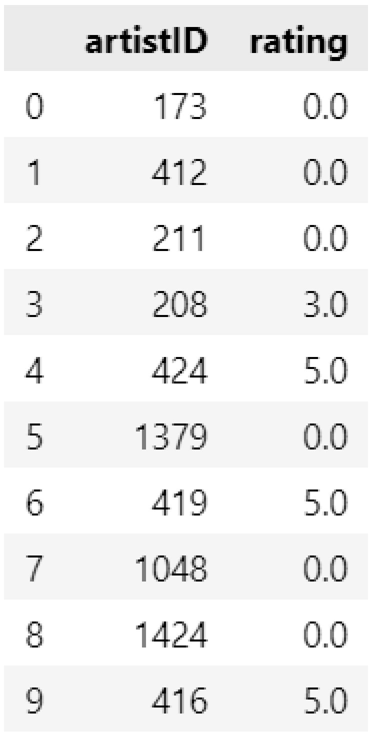


	artistID	rating
0	154	2.0
1	159	3.0
2	196	5.0
3	199	5.0
4	200	3.0
5	208	3.0
6	217	4.0
7	219	3.0
8	225	5.0
9	228	3.0

Following this we have the merge of the recommended artists and the artists the user have already rated.



```
# This is the merge of the recommended artists by our model and the artists that user have already rated.
merged = recom_cb.merge(recom_grpd, how = 'left' , on= 'artistID')
merged.fillna(0)
```
✓ 0.5s

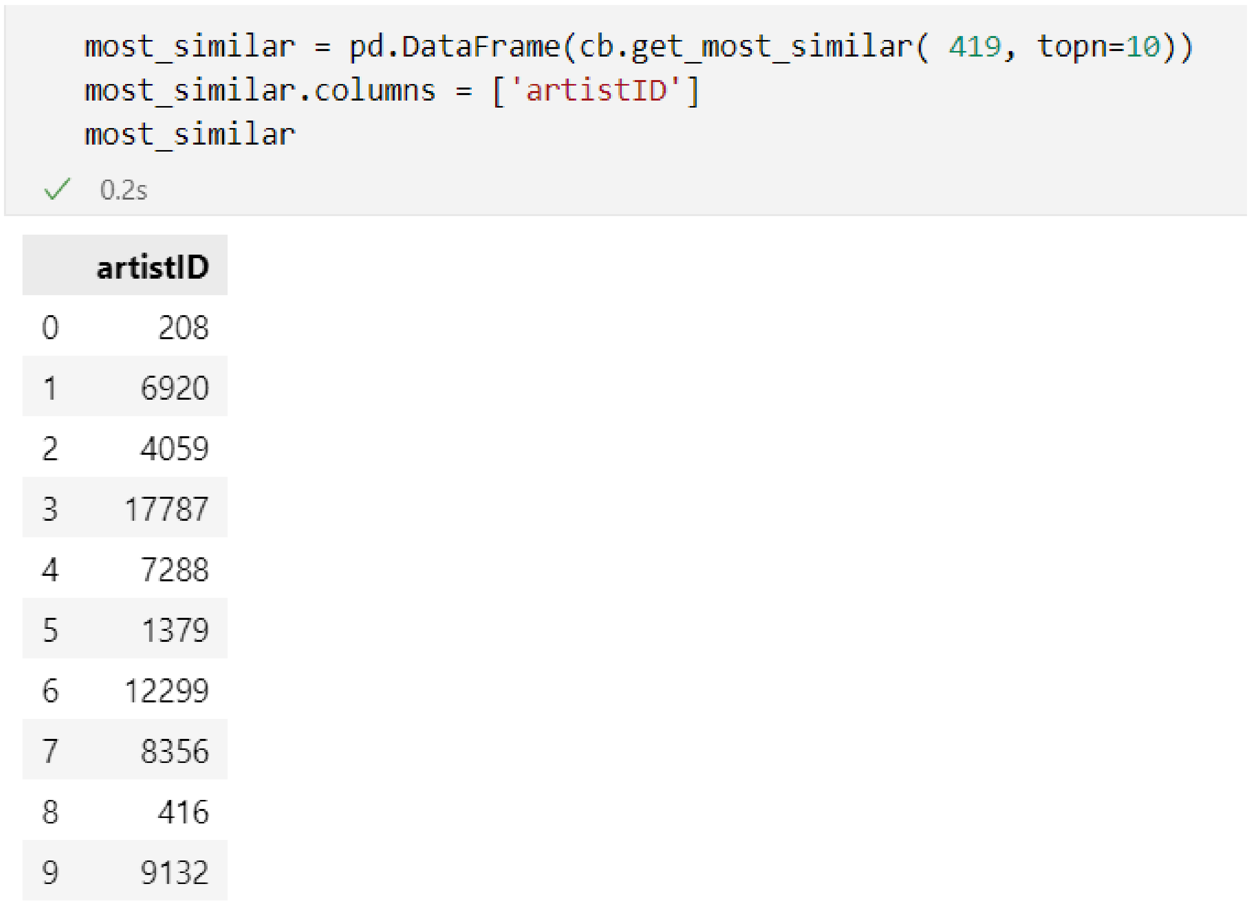


	artistID	rating
0	173	0.0
1	412	0.0
2	211	0.0
3	208	3.0
4	424	5.0
5	1379	0.0
6	419	5.0
7	1048	0.0
8	1424	0.0
9	416	5.0

- Here we can see, as per our Models Recommendations, we have 4 artists where the user have already rated (3 of them are 5 rating) and 6 artists are new recommendations.
- Artists for which the rating is 0 are the new Artists that our Model is recommending to the user.
- Recommending Three 5 rated artists suggests that the Model considers all the cases where, what customers likes and what new and similar artists we can Recommend to the Users.
- This also suggests that the model is recommending the artists to the users as we want it to be.
- So, the Recommendation strategy here is considering the artists which the user likes to listen while also suggesting the new and similar artists.

Finding out the Most similar Artists :

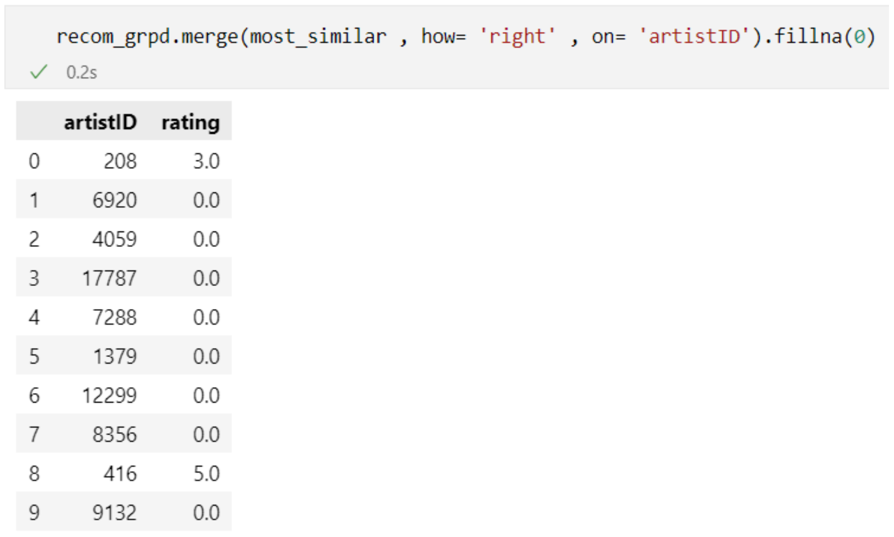
We are using this function cb.get_most_similar to find out the Most similar artists. this fuction takes the arguments artistID and topn.



```
most_similar = pd.DataFrame(cb.get_most_similar( 419, topn=10))
most_similar.columns = ['artistID']
most_similar
```

✓ 0.2s

	artistID
0	208
1	6920
2	4059
3	17787
4	7288
5	1379
6	12299
7	8356
8	416
9	9132



```
recom_grpd.merge(most_similar , how= 'right' , on= 'artistID').fillna(0)
✓ 0.2s
```

	artistID	rating
0	208	3.0
1	6920	0.0
2	4059	0.0
3	17787	0.0
4	7288	0.0
5	1379	0.0
6	12299	0.0
7	8356	0.0
8	416	5.0
9	9132	0.0

- Here this suggests that in the most similar artists also we have 2 artists who was already rated by the user the rest all are like the user we are recommending the artists.

Conclusion

- To conclude, the platforms use many tools that allow them to make the best possible recommendations to their users. And to stay in the competition, it is very important to constantly improve the personalization of recommendations to generate customer engagement and not to go to competitors.
- Today, it's good you have an interesting recommendation system that recommends the top 10 most listened to music on the platform, but it's not enough to stay in competition with competitors.
- Therefore, we have offered you different algorithms during this study to personalize the recommendations as much as possible and to encourage users to listen to more varied artists. These recommendations will literally create a strong bond with the customer who will be delighted to have personalized recommendations and will therefore listen to more new music.
- Recommender systems have an important place in marketing. Your expenses in these tools will quickly be transformed into sustainable investments and this will differentiate you from your competitors. In addition, it can make life easier for customers by saving them time.
- None of the algorithms presented in the study can answer all the problems, because the methods have different advantages and disadvantages depending on the marketing issues. However, it will be necessary to be vigilant about the implementation of these algorithms because this requires the collection of relevant data, with qualitative data, the construction of profiles, the selection of the right products.

Reference

Prototyping a Recommender System Step by Step Part 1: KNN Item-Based Collaborative Filtering | by Kevin Liao | Towards Data Science

Matrix co-clustering can reveal the latent structure. Discovered 'white... | Download Scientific Diagram (researchgate.net)

https://analyticsindiamag.com/a-guide-to-building-hybrid-recommendation-systems-for-beginners/

https://towardsdatascience.com/recommendation-systems-explained-a42fc60591ed

Coursework

StackOverflow

YouTube Videos for few concepts

THANK YOU