GROUP PROJECT

RECOMMENDATION TOOLS 2022

BY
SWASTHIK VELLINGIRI KOWSALYA
HARIKRISHNAN GOPALAKRISHNAN

Business Problem

Description:

One of the leading music streaming services, LastFM, plans to improve its recommendation system by providing its customers with a better user experience and more streaming time on the platform. We were asked to improve the currently existing system, which recommends ten famous artists to all platform users.

Objective:

The business objective is to build a better recommendation system that outshines the existing recommendation system in recommending the artists to the users. The final selected recommendation model should encourage users to engage with a variety of different artists to increase the overall streaming time on the platform.

Dataset Description:

We have four datasets-**Artists.dat,tags.dat,user_artists.dat and user_taggedartists.dat.**

Variables available within these datasets include :

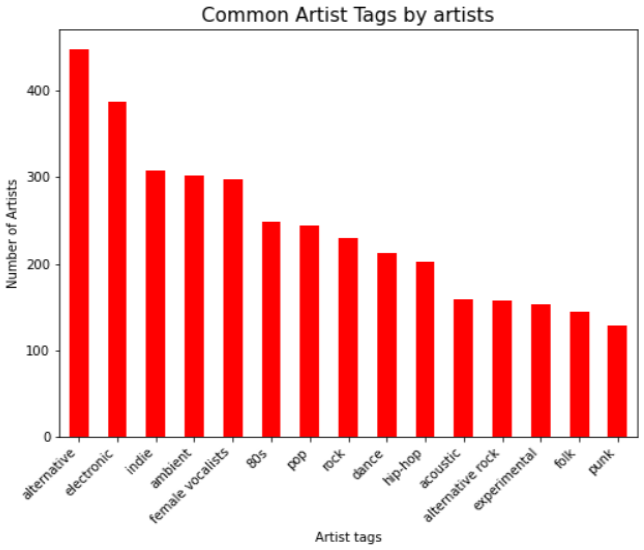
artistID - unique identifier of artists
name - Name of the artist
url - Link to the LastFM artist profile
Pictureurl - Link to the artist's profile photo
tagID - unique identifier of tags
tagValue - Tag Description
userID - unique identifier of users
Weight - streams or play count
Day - day the user tagged the artist
Month - month the user tagged the artist
Year - year the user tagged the artist

These variables are preprocessed and used to build different models for recommendation systems.

Exploratory Data Analysis - EDA

The exploratory data analysis is done to get an overview of the data and determine the interesting relations among the variables in the different tables.

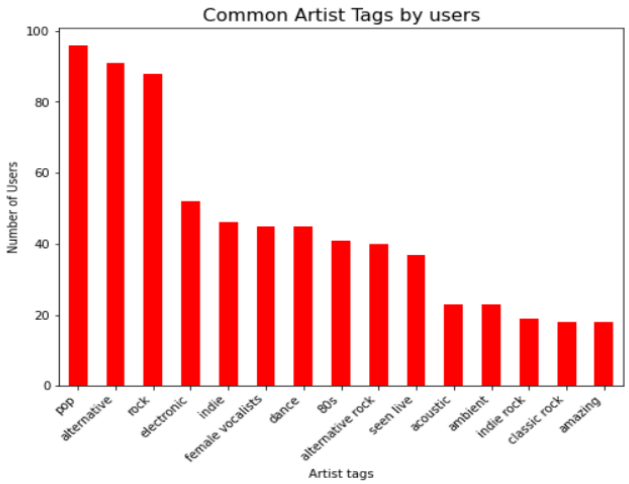
Common artist Tags by artists



To determine the most common tags with respect to the Artists, we merged the tables user_taggedartists and tags. This gives a table that contains information about the UserID,TagID and its Tagvalue.The artistID is grouped and the count of the tagvalue is considered in this case.

From the above plot,we can observe that the “Alternative” Artist tags are the most common artist tags with more than 400 artists and followed by “Electronic” Artist tags with almost 400 Artists.

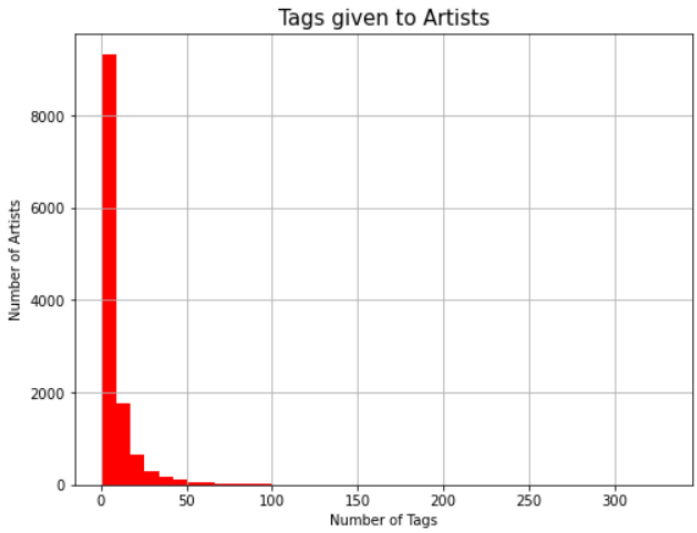
Common artist Tags by Users



To determine the most common tags with respect to the users,we merged the tables user_taggedartists and tags.This gives a table which contains information about the UserID,TagID and its Tagvalue. The userID is grouped and the count of the tagvalue is considered in this case.

From the above plot,we can observe that “Pop” is the most common Artist tag and popular amongst almost 100 users and followed by “Alternative” and “Rock” with 90 users. Though Artist tags “Pop” and “Rock” have relatively few artists ,it has huge fandom amongst the LastFM platform users.

Tags given to Artists

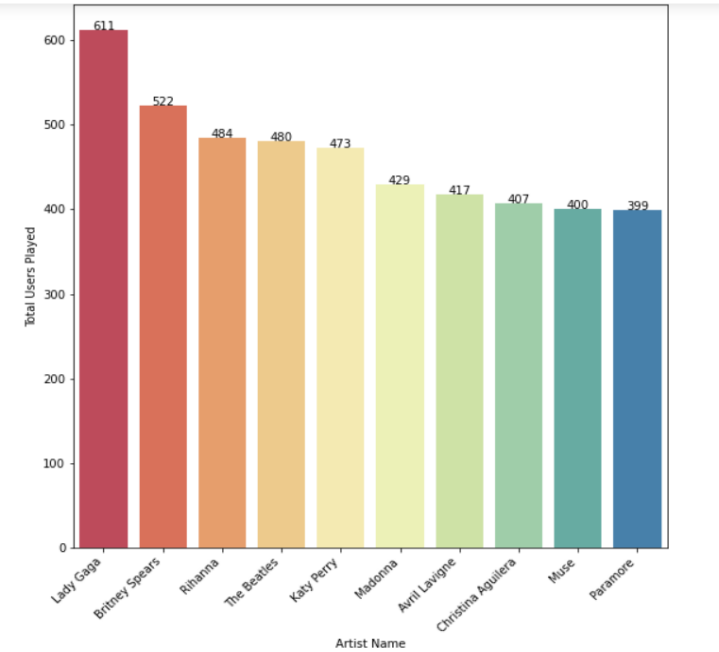


To show the relation between the artist and artist tags,we grouped the artistID and the unique Tag value is considered from the user_tagged artists table.

From the above plot,we can observe the distribution of user tags with respect to the number of artists. The distribution is skewed i.e Longtail.so, the majority of artists compose only a few music genres (tag value).

Artists with more users

In order to spot the most popular artists in terms of users, We have created a separate table by merging the artist and user_artists tables. The artist names are grouped, count of userID and sum of weight (Play count) are considered.



In this case, the top 10 artists with more users are shown in the table and plot.From the plot, We can observe that Lady gaga is the most popular artist with 611 users followed by Britney spears with 522 users and Rihanna with 484 users.

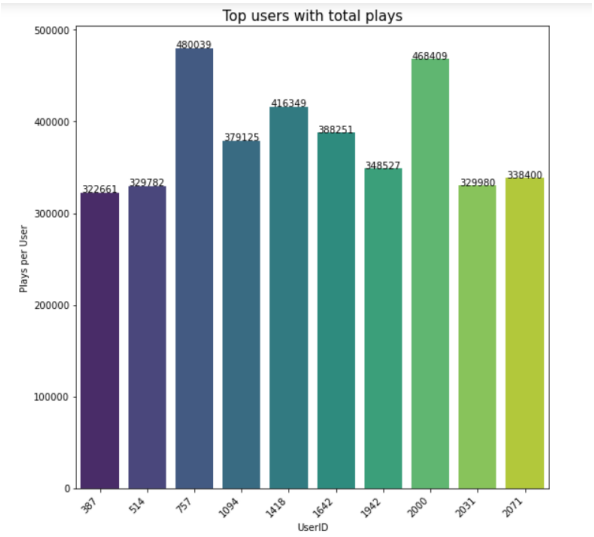
name	Total_num_users	Total_num_A.plays	Avg_user_plays
Britney Spears	522	2393140	4584.559387
Depeche Mode	282	1301308	4614.567376
Lady Gaga	611	1291387	2113.563011
Christina Aguilera	407	1058405	2600.503686
Paramore	399	963449	2414.659148
Madonna	429	921198	2147.314685
Rihanna	484	905423	1870.708678
Shakira	319	688529	2158.398119
The Beatles	480	662116	1379.408333
Katy Perry	473	532545	1125.887949

In the table, we can see that Though the Lady gaga is streamed by more users, the number of plays and average user plays is lesser when compared to other artists.

Users with total plays or streams

In order to spot users who use the platform very often, We have created a separate table by merging the artist and user_artists tables. The users are grouped, count of artist names and the sum of weight(Play count) are considered.

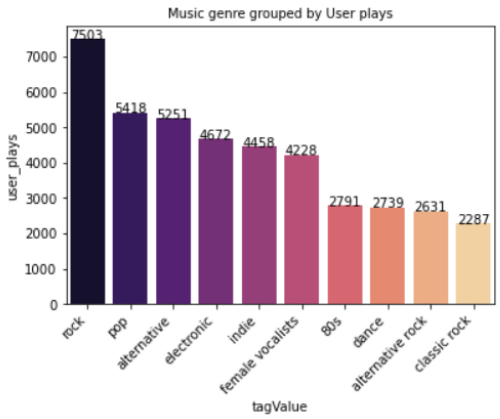
userID	Total_num_artists	Total_num_U.plays
757	50	480039
2000	50	468409
1418	50	416349
1642	50	388251
1094	50	379125
1942	50	348527
2071	50	338400
2031	50	329980
514	50	329782
387	50	322661



In this case, the top 10 users with more plays are shown in the table and plot. From the plot, We can observe that userID 757 is the highest with 480039 streams and followed by userID 2000 with 468409 streams.

Music Genre Popularity

tagValue	user_plays
rock	7503
pop	5418
alternative	5251
electronic	4672
indie	4458
female vocalists	4228
80s	2791
dance	2739
alternative rock	2631
classic rock	2287



To identify the Music genre's popularity in terms of the number of user plays, we have created a table by merging the tags,user_TaggedArtists, and Artist tables. The Tag values are grouped and the count of userID is considered.Table is sorted by the userID count in descending order.

From the Plot, We can observe that the music genre “Rock” is the most popular genre with user plays of 7503 and followed by “pop” with 5418 user plays. We have already seen a case where the music genre - “Pop” and “Rock” has more users on the platform. So this information from the plots is coherent.

Pre-Processing Steps

General Pre-Processing:

- The data in the weights column of the user_artists.dat table is continuous and skewed. So, it is categorized by using a Quantile-based discretization function (qcut) to indicate the quantile membership for each data point.
- Created a content Table by merging user tagged artists table and the tags table.
- Checked if all the userID and artist ID are in the user artists table and the content table. The missing artist IDs were added to the content table.
- Since the tags of each artist were distributed in various rows, they were concatenated such that there is one row per artist in the content table.
- Removed the duplicates in the content table.

Preprocessing for Content Based Filtering:

- The tags column in the content table was tokenized, and it was filtered only for alphabets and converted to lowercase.
- Removed the stop words and stemming was performed.
- TFIDF vectorizer was performed to the tags column in order to convert document-term-matrix to the sparse format.
- Convert the matrix to dataframe with artist ID as index.

Train Test Split:

- To train and test the model ,we use the data with columns userID,artistID and weight1 (categorized using qcut function) from the user_artists table. The data is

split into three chunks. One is the train which contains 70% of the data and the remaining 30% is split into two test sets one is for the models and one evaluation set is to evaluate the linear regression and random forest hybrid models.

- The train, test and evaluation set is converted into the surprise package format before building the model. The reader rating scale ranges from 0 to 9 based on the qcut function precision we chose.

Building Recommendation Models

There are four collaborative models built, one content based model and 4 hybrid models built for this LastFM dataset. The descriptions and the parameters selected for these models are posted below.

The models built are Content based filtering, BaselineOnly, KNNbasic, ALS, SVD, SVDpp and four hybrid models.

Content Based Filtering

Description:

Based on the similarities found in the features of the items each user has rated, content-based filtering provides recommendations. In other words, it understands and learns the features of the items each user has rated.

In this model, the tags of the artists are used as a content and a word matrix is created mathematically using the TF-IDF method. The word matrix is then used by the content based function with the help of the nearest neighbors to predict the artists based on the tag values.

Collaborative Filtering

Collaborative filtering is the most prominent recommendation approach that can filter out items that a user might like on the basis of reactions by similar users.

The first step towards building a recommendation system that can recommend items to users based on their preferences is to establish similar users and items. The next step involves predicting the ratings of items that have not yet been rated.

Collaborative filtering has a range of algorithms that can be used in multiple ways to find similar users or items and multiple ways to calculate ratings based on ratings of similar users.

The metric used to measure the accuracy of the algorithm result is the Root Mean Square Error (RMSE), in which you predict ratings for a test dataset of user-item pairs whose rating values are already known. The difference between the known value and the predicted value would be the error value. First, find the average or mean for Square of all the error values of the test set, and then take the square root of that average to get the RMSE.

In our case, the base matrix is created with columns UserID, ArtistID, and weight from the table- user_artists.dat for the collaborative filtering approach.

1. Item-based CF

Description:

The item-based collaborative filtering technique filters out similar items based on items that users have liked or interacted with positively in the past.

In our case, it filters out similar artists based on artists that users have liked or streamed music of that particular artist many times in the past. Two popular similarity measure in item-based CF:

Pearson correlation- It measures the strength of the linear relationship between two variables-artists and weights.

Cosine similarity-It views two artists and their weights as vectors and defines the similarity between them as the angle between these vectors.

Fitting the Model: KNNbasic()

The surprise format pre-processed data is fitted using the following parameters :
k – The (max) number of neighbors to take into account for aggregation.(Default = 40)

min_k – The minimum number of neighbors to take into account for aggregation. If there are not enough neighbors, the prediction is set to the global mean of all ratings.(Default = 1)

sim_options – A dictionary of options for the similarity measure. See Similarity measure configuration for accepted options.

Example: options = {'name':'Pearson', 'user_based':False}

Pearson- Pearson correlation measure.

-The model is best fitted using the **parameter tuning technique - GridsearchCV** from suprise.model_selection package.

Accuracy Metrics:

RMSE: 2.3028
The other measures are shown separately in the evaluation measures comparison with other models.

Advantages:
Less scalability issues when compared to user-based approach
Disadvantages:
High sparsity leads to few common ratings between two users

Matrix Factorization¶

Matrix factorization is the collaborative-based filtering method where matrix m*n is decomposed into m*k and k*n. It is basically used for the calculation of complex matrix operations. Division of matrix is such that if we multiply factorized matrix we will get original matrix It is used for discovering latent features between two entities.

In the case of the LastFM platform, the matrix is used for identifying the latent features between the entities-artists and weights.

2. Alternating least squares(ALS):

Description:

ALS algorithm works by turning the user-item interaction matrix into two rectangles of lower dimensions. The user matrix can be found by viewing the columns as latent factors, and the rows as users. The other matrix is the item matrix where rows are latent factors and columns represent items.

The algorithm's baselines are calculated by calling the **bsl_options** parameter. This is a dictionary of options that uses the key **'method'** to identify the method to use. Values accepted are **'als'** (default) and **'sgd'.** Other options may also be set based on their value.

Fitting the Model: BaselineOnly()

The surprise format pre-processed data is fitted using the following parameters :

"method": "als"-(default) for Alternating least-squares approach .

'reg_i': The regularization parameter for items. Corresponding to λ2. (Default = 10).

'reg_u': The regularization parameter for users. Corresponding to λ3. (Default = 15).

'n_epochs': The number of iterations of the ALS procedure. (Default = 10).

-The model is best fitted using the **Parameter tuning technique-GridsearchCV** from suprise.model_selection package.

Accuracy Metrics:
RMSE: 1.6557
The other measures are shown separately in the evaluation measures comparison with other models.

Advantages:
-Minimize least-squares error of the observed ratings

Disadvantages:
-Transformed data after matrix factorization may be difficult to understand.

3. Singular Value Decomposition(SVD):

Description:

This technique works in such a way that reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where K<N).

The matrix factorization is done on the user-item rating matrix. From a high level, matrix factorization can be used to determine the 2 matrices whose product is the original matrix.

Fitting the Model:SVD()

The surprise format pre-processed data is fitted using the following parameters :

n_factors – The number of factors. (Default = 100).

n_epochs – The number of iterations of the SVD procedure. (Default = 20).

biased (bool) – Whether to use baselines (or biases). (Default = True).

random_state – Determines the RNG that will be used for initialization. (Default = None).

Accuracy Metrics:
RMSE: 2.3028
The other measures are shown separately in the evaluation measures comparison with other models.

Advantages:
-Captures important factors/aspects and their weights
-Improves the metrics by minimizing the least square errors.

Disadvantages:
-Transformed data may be difficult to understand.

4. Singular Value Decomposition Plus-Plus (SVDPP)

This algorithm is an extension of the SVD algorithm which takes **implicit ratings** into account.

Fitting the Model:SVDPP()

The surprise format pre-processed data is fitted using the following parameters :

n_factors – The number of factors. (Default = 100).
n_epochs – The number of iterations of the SVD procedure. (Default = 20).
biased (bool) – Whether to use baselines (or biases). (Default = True).
random_state – Determines the RNG that will be used for initialization. (Default = None).

Accuracy Metrics:

RMSE: 1.6835
The other measures are shown separately in the evaluation measures comparison with other models.

Advantages:
-Captures important factors/aspects and their weights
-Improves the metrics by minimizing the least square errors.

Disadvantages:
-Transformed data may be difficult to understand.

5. Co-Clustering

Description:

This algorithm does concurrent clustering of similar users and items based on the similarity of their pairwise interactions and It iteratively computes the user-item rating estimates for each cluster.

In the case of the LastFM platform, the clustering is done based on similar users listening to the music of the same artists, and the cluster is estimated with a similar rating value after computation.

Fitting the Model:CoClustering()

The surprise format pre-processed data is fitted using the following parameters:

n_cltr_u (int) – Number of user clusters. Default = 3.

n_cltr_i (int) – Number of item clusters. Default = 3.

n_epochs (int) – Number of iterations of the optimization loop. Default = 20.

random_state – Determines the RNG that will be used for initialization. (Default = None).

Accuracy Metrics:

RMSE: 2.0792
The other measures are shown separately in the evaluation measures comparison with other models.

Advantages:
-Iterative computation will re-assign clusters in order to minimize squared errors.

Disadvantages:
-Original matrix must be fully-dense (no missing values allowed).

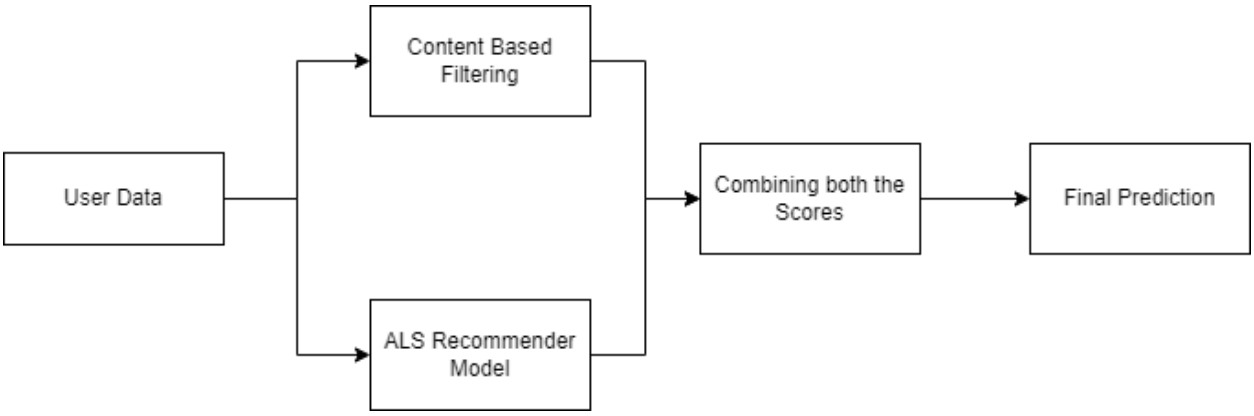
Hybrid Models

The Hybrid Model used in this project is the weighted one. In the Weighted Hybrid Model, predictions of two recommender algorithms are weighted to produce one final prediction. Usually, a content based filtering will be combined with a collaborative filtering system. So, in this project, a content based system is combined with collaborative systems such as ALS and SVDpp separately. Hybrid Models considered in this project are,

1. Content Based Filtering + ALS (Basedline)
2. Content Based FIltering + SVDpp
3. Linear Regression Prediction of Hybrid 2 (CB + SVDpp)

4. Random Forest Prediction of Hybrid 2 (CB + SVDpp)

Hybrid Model 1: Content Based + ALS (Baseline)

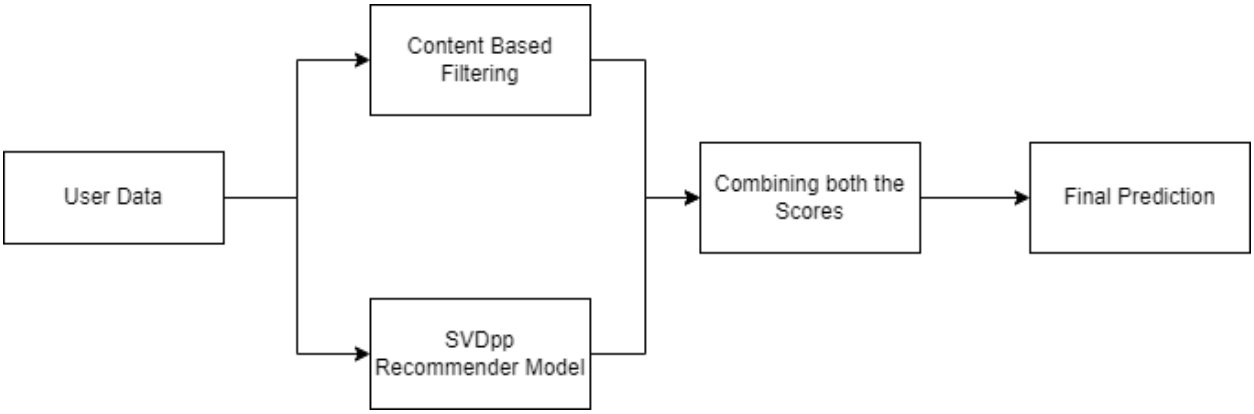


Process:

In this model, the estimated scores calculated using the content based filtering and ALS which are explained above are taken into account. Then an average is taken between these two scores for a particular user. The average score is considered as the final prediction.

Hybrid Model 2: Content Based + SVDpp

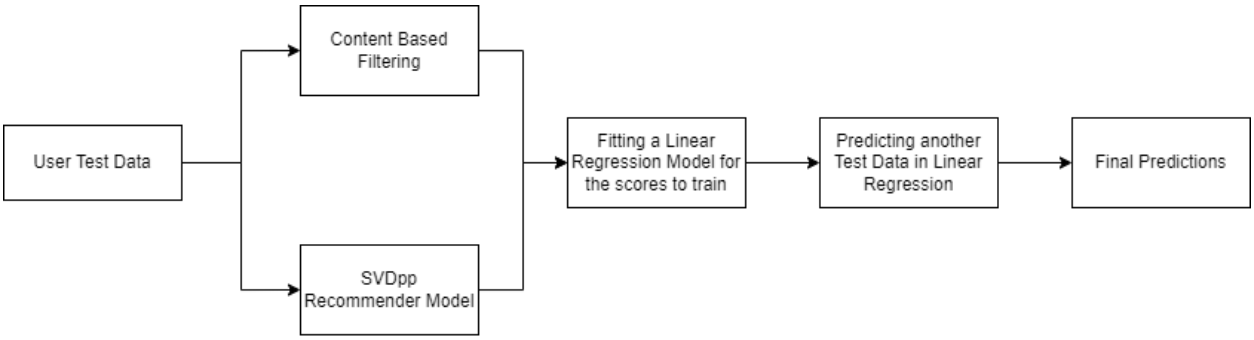
In the second Hybrid Model, a content based filtering model is combined with another collaborative filtering model called SVDpp.



Process:

The Process is the same as the previous hybrid Model and the only difference is that this hybrid model has a different collaborative filtering model. Again, the average of the two models, Content based and SVDpp models are taken and considered as the final prediction.

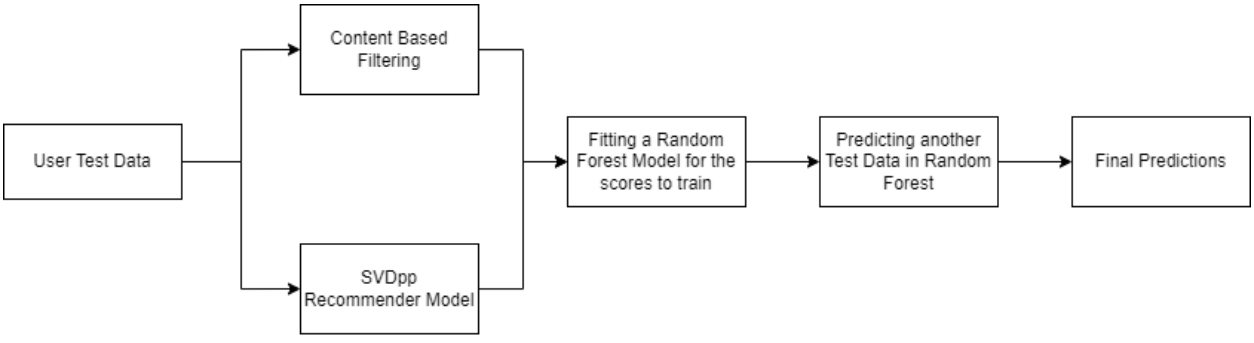
Hybrid Model 3: Linear Regression Prediction of Hybrid 2 (CB + SVDpp)



Process:

This process is quite similar to the conventional hybrid process. But instead of taking weighted averages between the predicted values of two different models, they are used to train a linear regression model. Then using this linear regression, the estimated scores of a different test dataset is predicted and evaluated.

Hybrid Model 4: Random Forest Prediction of Hybrid 2 (CB + SVDpp)



Process:

This process is similar to the Linear regression model, but instead of using a linear regression, a random forest machine learning model is used to train the predicted values from two different recommender systems, which are usually one content based and one

collaborative. Again, a different dataset is tested for predictions in the random forest model and it is evaluated.

Advantages of Hybrid Recommender systems

1. Hybrid systems can result in more accurate predictions.
2. Hybrid systems can overcome not enough data for collaborative models by inducing the content based systems thereby increasing the accuracy.

Cross Validation and Evaluation of the Models

Cross Validation is an important step in assessing the quality of the model selected by comparing it with the other models. In the project, a cross validation step was executed to choose the best performing models to further test them by splitting the dataset into train and test sets. The initial cross validation included comparing the RMSE scores and MAE scores of the following models.

1. KNNBaseline()
2. KNNBasic()
3. BaselineOnly()
4. SVD()
5. SVDpp()
6. CoClustering()

The cross-validation process was executed using the “cross_validate” function present in the surprise library. The results of the cross validation process is presented below in the tabular format sorted on the best RMSE score.

Algorithm	test_rmse	test_mae
SVD	1.747679	1.397843
SVDpp	1.756474	1.358828
BaselineOnly	1.895470	1.597255
KNNBaseline	1.953434	1.577046
CoClustering	2.139262	1.669887
KNNBasic	2.654020	2.168501

From the above table, the models SVD and SVDpp are the best performing scores followed by BaselineOnly (ALS) in terms of RMSE scores. The cross-validation is also carried out by testing the models separately.

This evaluation step includes all the models used in the project including the content based filtering and the hybrid based models. Along with the RMSE and MAE scores, we also get to find the scores of other metrics such as Recall, Precision, F1 and NDCF@10. These scores are explained in detail for the model chosen in the next section.

	Content_Based	Item_Based	ALS	SVD	SVDPP_	Clustering	H:CB+ALS	H:CB+SVDpp	Hybrid+ LR	Hybrid: RF
RMSE	1.699809	1.932557	1.655654	2.178094	1.683452	2.079200	1.665838	1.656146	1.655865	1.665034
MAE	1.287678	1.463192	1.297248	1.605420	1.331483	1.609730	1.286720	1.289300	1.262929	1.257390
Recall	0.756789	0.759783	0.754830	0.596641	0.759267	0.673107	0.759203	0.767502	0.758099	0.756593
Precision	0.846582	0.823258	0.859378	0.875203	0.857443	0.827410	0.856219	0.856542	0.860154	0.861432
F1	0.799172	0.790248	0.803718	0.709562	0.805374	0.742325	0.804798	0.809581	0.805908	0.805616
NDCG@10	0.843000	0.822074	0.840814	0.840641	0.839084	0.829930	0.839167	0.837879	0.832921	0.850117

From the total evaluation of the models, we can see that the ALS model has the lowest RMSE score followed by Linear Regression Hybrid Model which is followed by Random Forest Hybrid Model. The hybrid models are performing better than the most other models for the selected dataset.

Evaluating Selected Model ALS (BaselineOnly)

Recommender systems are most commonly evaluated using RMSE and MAE scores. However, in this section, the selected model is evaluated based on RMSE, MAR, F1 and NDCG metrics. The evaluation scores of the ALS (BaselineOnly) model is as follows.

	ALS
RMSE	1.655654
MAE	1.297248
F1	0.803718
NDCG@10	0.840814

MAE:

The Mean Absolute Error is the average of the difference between the actual rating given by the user and the rating predicted by the recommender model. The negative scores are converted to positive by making the values absolute. The lower the MAE score is the better the predictions are. In the above ALS model, it is evaluated that the model has a MAE score of 1.297 which is quite good for a data with rating scale 1 to 10.

RMSE:

This model is selected as the final model since the RMSE score of this model is the lowest of all the models. RMSE abbreviated as Root Mean Square Error is calculated by taking a square root of the square of the difference between the actual rating given by the user and the rating predicted by the recommender model. RMSE is similar to MAE but instead of using absolute values to remove negative scores, RMSE squares the difference and takes a square root of the whole term. The RMSE score evaluated for the ALS model is 1.655

F1 Score:

In order to understand and obtain the F1 score, we need to understand two different decision support metrics. First is Precision and the second is Recall.

Precision in simple terms, is the number of selected items that are relevant. For example if the recommender system selects 10 items out of which 5 items are correct and relevant, then the score for precision will be 50%. In the ALS model, the precision score is 0.859 which means if the recommender selects 10 artists, 8.5 items will be artists.

Recall is similar to Precision but it calculates based on the number of relevant items that are recommended. For example, if there are 10 relevant items and if the recommender selects 8 items, then the recall will be 80%. In the ALS model, the recall score is 0.75 which means if there are 10 relevant artists, the recommender selects 7.5 artists from them.

Now that we know what Recall and Precision is, the F1 score can be calculated. F1 score is calculated by taking the harmonic mean of Precision and Recall. The f1 score can be expressed in the following way.

$$F1\ score\ =\ 2/((1/Precision)+(1/Recall))$$

The f1 score for the ALS model is 0.80 or 80% which is pretty good.

NDCG score:

NDCG can be abbreviated as Normal Discounted Cumulative Gain. In order to understand NDCG, it is important to know its predecessors.

Cumulative Gain is the sum of all relevant recommendation scores in a recommendation set. Since cumulative Gain does not consider the position of the relevance scores in a set, we will need to discount the relevance score by dividing it with the log of the position. This process is called Discounted Cumulative Gain. This Discounted Cumulative gain will not be the same for every user, and hence to get a single value for the model, average of these scores are taken to find a final score. Hence this process is called Normalized Discounted Cumulative Gain.

It is important to recommend artists that are highly relevant to the user rather than moderately relevant. Hence it is important to measure this NDCG rating. The NDCG score for the ALS model is 0.84 or 84% is a good score.

Sample Recommendations for a User:

Based on the ALS model, the recommendations for the users are predicted. For a user whose ID is 2, the following are the predicted artists for the user.

	ALS
0	Tommy Lee
1	The Tiger Lillies
2	Renegade Five
3	Evading Downfall
4	Whitechapel
5	I Killed the Prom Queen
6	Jacques Brel
7	Tegan and Sara
8	Shaaman
9	Blur

These recommendation predictions are made using a custom function which checks the estimated predicted values of the ALS model to provide predictions. From the result, we can see that Tommy Lee is the highest relevant artist for this particular user followed by The Tiger Lillies, Renegade Five etc. Recommending these artists from top to bottom will definitely help the user spend more time on the platform and thereby increase the revenue for the company.

If we compare it with the top 10 most popular artists (Table below) based on the play count, not even one of the recommendations given to this particular user is present in the top 10. Hence it is proved that this model predicts recommendations that are aligned with the user's interest rather than just recommending top 10 users. The top 10 artists by play count are displayed below for reference.

name	Total_num_users	Total_num_A.plays	Avg_user_plays
Britney Spears	522	2393140	4584.559387
Depeche Mode	282	1301308	4614.567376
Lady Gaga	611	1291387	2113.563011
Christina Aguilera	407	1058405	2600.503686
Paramore	399	963449	2414.659148
Madonna	429	921198	2147.314685
Rihanna	484	905423	1870.708678
Shakira	319	688529	2158.398119
The Beatles	480	662116	1379.408333
Katy Perry	473	532545	1125.887949

Reason Behind the Recommendations

The collaborative filtering algorithms work based on the historical data of the other users who are having similar interests. The concept is explained in simple steps below.

Step 1: An User named "A" listens to a particular song from an artist.
Step 2: The system compares this user A's artist and interests with all other users in the network and identifies other users with similar interests.
Step 3: The system recommends the artists streamed by the other similar users to "A".

The recommendations for a particular user identified using the ALS algorithm can be suggested to the users using the tagline, “Users who streamed this artist, also streamed these”. The topmost recommendation for this user was Tommy Lee and the reason this artist has the highest relevance is that the user number 2 had streamed songs which were streamed by other users who had also streamed Tommy Lee’s songs. A number of other similar users streamed Tommy Lee’s songs than the Tiger Lillies songs. And hence the Tiger Lilles are placed second and Tommy Lee is placed first.

Strategies to Diversify Artist Recommendations

While LastFM currently has implemented top 10 artists recommendations for all the users, it is important to diversify the artists. Using the above-mentioned algorithms will help the lastFM diversify the artists among the users. However, there are slight setbacks in these models as well which need to be treated to completely diversify the recommendations. Some of the limitations and strategies that can be followed are posted below.

1. If a new artist is added to the data, it is most unlikely that the new one will get recommended for any user. So the LastFM could boost the ratings or weights of such new artists or underrated artists if found, manually so that the recommender systems pick them.

2. Most often, the users get to listen only to the popular artists. Popular artists get more play count naturally. So the ratings should not be only based on weights. LastFM can also consider adding other features to increase the ratings of the artists. Some examples can include Awards received, currently trending on social media, etc.

3. Creating a like or upvote option for artists could help gather more data and recommend users in a better way than just recommending based on a song played. In most cases, if a new user listens to a song, they may or may not like it. It won't be correct to recommend the similar artists based on the song they did not like. So it is a must to understand whether the user likes a particular artist before we recommend them.

Conclusion

The recommendation systems are gaining popularity across various industries. Particularly in the online streaming companies, it is important to value customers' time and effort and have them stay in the portal for a longer period of time. Though all the recommendation algorithms provide recommendations, it is important to identify how it works and what would be suitable for the particular industry based on the data available. This project has covered popular recommender systems and evaluated the best model identified using cross-validation.

References:

1. https://machinelearningmastery.com/k-fold-cross-validation/
2. https://www.datacamp.com/community/tutorials/recommender-systems-python
3. https://www.geeksforgeeks.org/recommendation-system-in-python/
4. https://realpython.com/build-recommendation-engine-collaborative-filtering/
5. https://towardsdatascience.com/hands-on-content-based-recommender-system-using-python-1d643bf314e4
6. https://medium.com/analytics-vidhya/content-based-recommender-systems-in-python-2b330e01eb80
7. https://medium.com/fnplus/evaluating-recommender-systems-with-python-code-ae0c370c90be
8. https://github.com/cipher813/recommender_system/blob/master/notebooks/01_LastFM_CollaborativeRecommender.ipynb
9. https://stackoverflow.com/questions/43214978/seaborn-barplot-displaying-values
10. https://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html