**Capstone Project: Song Recommendation System for Spotify**

**Final Report**

# **Executive Summary**

This project proposes a model for recommending songs to users of Spotify music app. Several popular recommendation models have been developed in this project. Each of the models developed is further tuned to increase accuracy. The results from the finetuned model are compared to propose the final model.

Models are evaluated based on following metric (see Appendix for more information)

* What percentage of recommendations made are relevant (Precision). Low precision will lead to low customer satisfaction and losing the customer.
* What percentage of total relevant items are recommended. Too many false negatives will lead to not recommending items liked by the user. This would lead to low customer satisfaction and losing the customer
* F\_1 score – Harmonic mean of Recall & Precision

## Key Takeaways

* The proposed solution is a rule-based approach.
* The predictions are based on user/song interaction and some basic information on songs (title, album, artist names, release year).
* Results & accuracy of model are dependent on data quality, this is to say that the results presented here consider the information available from data.
* Users with less than 9 neighbors, song recommendations would be based on global average.
* Recommendations for new users would improve as they interact with more songs.
* Increase in user satisfaction by implementing recommendations would reduce costs associated with customer retention.
* Satisfied customers would increase app popularity by word of mouth thus decreasing cost of acquiring new customers.

## Next Steps

* Recommended model can be moved to production.
* Handed over the model to ML operations to maintain in production.
* Model to be re-trained at intervals by ML Ops when more user/item interactions are available.
* Collect more informative data to improve model recommendations

# **Problem Summary**

We aim to address following problem statements from this project -

* Providing relevant recommendations as per user’s existing likings as well allowing them to explore new songs.
* Recommending most trendy songs to the new & existing users.
* Addressing cold start problem.

# **Solution Summary**

The recommendations model is based on the ratings given to songs by users. However, the ratings data is not available, so the listening count has been inferred as user rating for model building. The rating range is set at 1 – 5.

The higher the listening count would mean the higher the song would be rated by the user. To address the biasness caused by songs which are listened to by users hundreds/thousands of times the listening count is flattened to max rating 5. This would mean that the song listened to 5 times is equally liked to a song listened to 100 times.

The threshold for a song to be relevant is set at 1.5, meaning a song with rating prediction greater than equal to 1.5 is a relevant recommendation. The rating distribution has formed the basis for selecting this threshold. The rating distribution (Image 1) chart is available in the appendix section.

In addition, features of songs have been used for song recommendations. The title, artist & album name of the song is used as information to find similar songs for recommendation.

The data is split into training and test data, model is trained on training data and for model evaluation test data set has been used.

Users with less than 90 interactions have been dropped from the available data. As well songs with less than 120 interactions with users have been dropped. This helps us to keep the data informative for model building. To understand this better, assume a user ‘A’ has listened only one song, confidence in similarity of user A with other users based on only one song would be low. Thus, the predictive power of recommendation would be weak.

The suggested rule-based recommendation system is as follows:

1. Cold Start problem:
   1. New user with no data on song interactions available, **Popularity Based Recommendation** model to be used to recommend most popular songs to the user.
   2. For a new song released, **Content Based Recommendation** Model could be used which recommends similar songs based on the title, artist, and release of the song.
2. For existing users with some data available on song interactions, song recommendations is based on the **User-User Similarity Optimized** model. The model recommends songs based on min 9 closest neighbors and max 30 neighbors. The model will calculate the rating of songs in the database and suggest songs in the decreasing order of the rating.

The goodness of the selected model is gauged from Precision and Recall of the model on test data. For the proposed model, following are the results:

Precision of the model is 46.1%. This means that out of 100 songs recommended by the model, 46% of the songs are relevant

Recall of the model is 83.1%. This means that out of the total relevant songs 83.1% songs are recommended by the model.

## Model Comparison

For the purpose of finding the best model, the available user/song interaction data is interpreted as follows (explained in option A &B below). The results from the 2 strategies have been compared, which is presented in the tabular format further in this section.

**Option A:** The strategy is to keep the relevant data to increase the recommendation power of the model. In this effort, the number of plays greater than 5 has been flattened to 5 instead of dropping those interactions to prepare data for model. This means now that the interaction matrix has more data than strategy B below.

**Option B:** This approach has been taken to reduce the amount of data to put less load on computational resources. In this approach all the records where the playing count is greater than 5 have been dropped. This means the interaction matrix would be sparse and seemingly adverse effect is that in this process, we have dropped relevant user-item interaction. Obviously, dropping relevant information should lead to a weaker model, which is proved by the results in the following metric.

**Table Comparing results of models implemented under 2 approaches:**

**F\_1 Score** - is the basis for selecting the model as both recall and precision are important for song recommendations. As highlighted F\_1 Score for model User\_User Similarity optimized model is .593 which is highest and hence forms the proposed model.

**Recall** - for the model .831 is the highest. It means that 83.1% of relevant recommendations are made by the model. This means about 83% of the songs liked by users have been recommended by the model.

**Precision** - is quite small at .461 which is in the similar range for other models. This means that only 46% of the total song recommendations made are relevant.

**RMSE** - of the model is 1.2985 is higher than approach B (1.0521) for the same model. RMSE is high because the amount of data points in approach A is much higher compared to data points in approach B. As well note that RMSE for SVD model is lowest at 1.2224

Also note that, results for all models under approach A are better compared to Approach B. Thus, it gives us confidence to state that approach A has consistently performed better.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **Approach A** |  | **Approach B** |
| **Model Name** | **Model Evaluation Metric** |  |  |  |
| **User\_User Similarity Optimized Model** | **RMSE** | *1.2985* |  | *1.0521* |
| **Precision** | *0.461* |  | *0.413* |
| **Recall** | *0.831* |  | *0.721* |
| **F\_1 Score** | *0.593* |  | *0.525* |
|  | | | | |
| **Item\_Item Similarity Optimized Model** | **RMSE** | 1.2507 |  | 1.0328 |
| **Precision** | 0.464 |  | 0.408 |
| **Recall** | 0.742 |  | 0.665 |
| **F\_1 Score** | 0.571 |  | 0.506 |
|  | | | | |
| **Matrix Model - SVD Optimized** | **RMSE** | 1.2224 |  | 1.0141 |
| **Precision** | 0.468 |  | 0.415 |
| **Recall** | 0.772 |  | 0.635 |
| **F\_1 Score** | 0.583 |  | 0.502 |
|  | | | | |
| **Clustering Based Optimized Model** | **RMSE** | 1.3115 |  | 1.0689 |
| **Precision** | 0.461 |  | 0.402 |
| **Recall** | 0.649 |  | 0.559 |
| **F\_1 Score** | 0.539 |  | 0.468 |

# **Recommendation for Implementation**

## Key Recommendations

1. Implement strategy A to prepare data for the recommendations model.
2. The addition of more users and songs would require the model to be re-trained periodically to keep making relevant recommendations.
3. As more and more data is collected over time, data older than a particular date could be dropped to make the model more relevant to changing user preferences. This model would however have the potential to discard old items from recommendations.
4. More informative data must be collected to increase the accuracy and relevance of predictions. Specially, Precision of the model needs to be improved.

## Action Items

1. Recommended model can be moved to production.
2. Handed over the model to ML operations to maintain in production.
3. Model to be re-trained at intervals by ML Ops when more user/item interactions are available.

## Expected Costs

Following costs needs to be considered for implementing recommendation systems over time:

* The time component of model training would increase as more and more data is available to train the model.
* With increasing data, investment in hardware (storage capacity, processing power etc) would be needed that adds to the costs of model implementation
* Costs associated with collection and maintenance of more informative data about songs/users and interactions.
* Data privacy rules of the land need to be followed if more personal information on users is collected, as well it would require users to agree to data disclosure agreements.

## Potential Risks/Challenges

* Model suggestions are based on average rating of neighbors for the user. If the model could not find a minimum of 9 neighbors for the user, the predicted rating would be global average rating. In such a situation, there would be a drop in the quality of recommendation for the user.
* Need for model re-training to be evaluated based on how many new users join Spotify app and number of new items being released. Training the model too frequently would be computationally heavy. Alternatively training too little would lead to bad recommendations for the new users/songs.
* Songs which are not listened to by similar users would never be recommended to the user.
* The model does not consider changing user preferences as they age and could lead to similar recommendations repeatedly.

## Further Analysis

Precision of all the models is around 46%. If more informative data could be provided, the models could be re-evaluated for improving recommendations. For example, following additional data could be more informative

* Actual rating of song provided by user.
* Data on user preferences could be collected (genres, artists etc) to further personalize the recommendations.
* User information like age, gender, profession etc.
* Genres, lyrics of the songs.

# Appendix

Model Evaluation Metric

**Recall:** What percentage of total relevant items are recommended. The aim is to have low False negatives (alternatively high recall). Too many false negatives will lead to not recommending items liked by the user. This would lead to low customer satisfaction and losing the customer

**Precision**: What percentage of recommendations made are relevant. The aim is to lower the False Positives to reduce incorrect recommendations to the user. High False Positive (alternative low precision) would lead to low customer satisfaction resulting from incorrect recommendations. This would lead to a lost customer.

**F\_1 Score**: As both Recall & Precision are important metric, harmonic mean of Recall & Precision is used for model selection.

**RMSE**: This describes the error in the prediction of rating from the true rating

Image 1: Rating Distribution

Chart, bar chart

Description automatically generated

Image 2: Most popular songs by play count

Chart, bar chart

Description automatically generated

Image 3: Most popular songs by count of users interacted with song

Chart, bar chart

Description automatically generated