← Go Back to Capstone Project

Course Content

FAQs - Music Recommendation System

Why does GridSearchCV takes a long time to run?

GridSearchCV is used here to find the **best set of hyperparameters** among the given sets. This requires training the model with each possible set of hyperparameters and hence training multiple models. This results in taking more time to run the GridSearchCV. The time taken by GridSearchCV is also dependent on the number of hyperparameters. **The higher the number of hyperparameters, higher is the runtime.**

From the final dataset, why the songs with play_count more than 5 are removed?

In the final dataset, the number of songs with play_counts more than 5 is **extremely low,** which will result in an extremely sparse user-item interaction matrix. This will also increase the size of the matrix, which will make building a recommendation system model computationally expensive. Hence, all such songs are removed from the data.

How to merge the two datasets to prepare the final data for the recommendation process?

To merge the two constituent datasets in the project, i.e., "count_df" and "song_df", we need to first **drop the duplicate records** from the "song_df" and then merge it with the "count_df". It can be done with the help of the below code:

```
df = pd.merge(count_df, song_df.drop_duplicates(['song_id']), on="song_id", how="left")
```

This will create a dataset of unique user_id, song_id pairs.

How to resolve the RAM issues occurring while running the notebook?

First of all, it is needed to make sure that you are using Google Colab for this project. Now, if in google colab, you are facing any of the below issues in this project, then it is a problem with the filtering applied to the data.

a. Crashing of the Google Colab session
b. RAM issues
c. Runtime issues

The respective code for correct filters are as below (which are provided in the reference notebook):

The first filter:

```
RATINGS_CUTOFF = 90
remove_users = []
for user, num_ratings in ratings_count.items():
    if num_ratings < RATINGS_CUTOFF:
        remove_users.append(user)
df = df.loc[~df.user_id.isin(remove_users)]
```

The second filter:

```
RATINGS_CUTOFF = 120
remove_songs = []
for song, num_ratings in ratings_count.items():
    if num_ratings < RATINGS_CUTOFF:
        remove_songs.append(song)
df_final= df.loc[~df.song_id.isin(remove_songs)]
```

This shall resolve the issues.

I do not see user id 6958 in the final dataset. What to do?

The reason for this issue is the **incorrect filters** applied to the data. Applying the correct filters (refer to above question) will certainly include user id 6958 in the final dataset.

Why is label encoding needed in this project?

Label encoding is a method to convert categorical variables to numerical ones. In the current project, as the user_id and song_id are given in the text format, it becomes necessary to convert them to numbers so that we can pass them to the algorithm.

What is precision@k?

precision@k is a modified performance assessment metric for recommendation systems. Here, k is the number of items (songs here) recommended to a user.

For the present case precision@10 can be interpreted as the fraction of the 10 recommended songs that are actually liked/listened by the user. If the user listens to 6 songs then precision@10 will be 6/10, i.e., 0.6 (60%).

Is it allowed to add web scraped data to the original data of the capstone project?

Making changes to the original data is **not recommended** because it will alter the results and affect the novelty of the problem itself. Keeping this in mind, it is advisable to proceed with the given data only.

Happy Learning!

Previous

Next