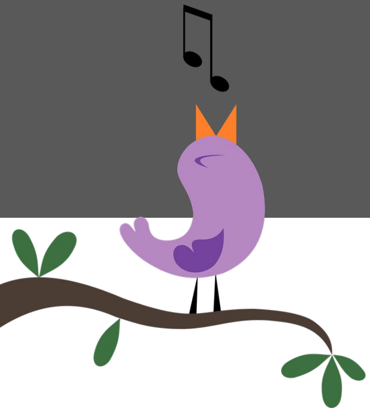
Great Learning
POWER AHEAD

Welcome! We will begin shortly

Music Recommendation System
Session - 1



Session Objectives and Agenda

1. The problem statement
2. Solution Approach
3. Best Practices
4. Question and Answers

The need for recommendation systems

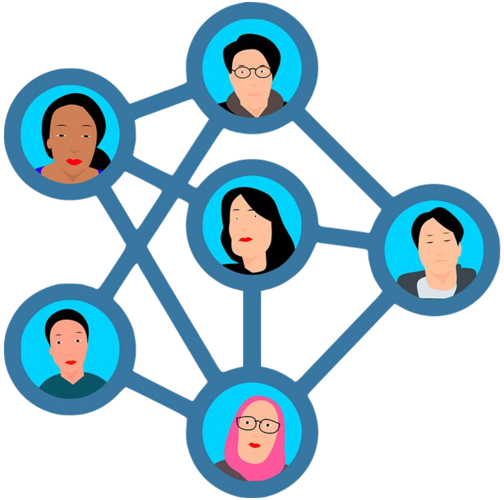
Recommendation systems are a **type of information filtering system** that use algorithms and statistical models to suggest items or content to users based on their preferences, interests, and behavior**. These systems aim to predict which items a user is most likely to be interested in, and recommend those items to the user.**

They help users discover new products, services, or content that they might be interested in based on their past behavior, preferences, and interests.

They optimize inventory management and supply chain operations by predicting customer demand and preferences and reducing waste and overstocking.

They improve customer satisfaction and loyalty by providing personalized recommendations and enhancing the user experience.

They increase sales and revenue by encouraging users to purchase more products or services and generating cross-selling and up-selling opportunities.



Context

Almost every internet-based company's revenue relies on the time consumers spend on its platform.

Spotify is one such audio content provider with a huge market base across the world.

With the ever-increasing volume of songs becoming available on the Internet, searching for songs of interest has become a tedious task in itself.

However, Spotify has grown significantly in the market because of its ability to recommend the **'best' next song to each and every customer based on a huge preference database gathered over time - millions of customers and billions of songs.**

This is done by using smart recommendation systems that can recommend songs based **on users' likes/dislikes.**

Objective

Build a recommendation system to propose the top 10 songs for a user based on the likelihood of listening to those songs.

Data Dictionary

The core dataset is the Taste Profile Subset released by The Echo Nest as part of the Million Song Dataset. There are two files in this dataset. One contains the details about the song id, titles, release, artist name, and the year of release. The second file contains the user id, song id, and the play count of users.

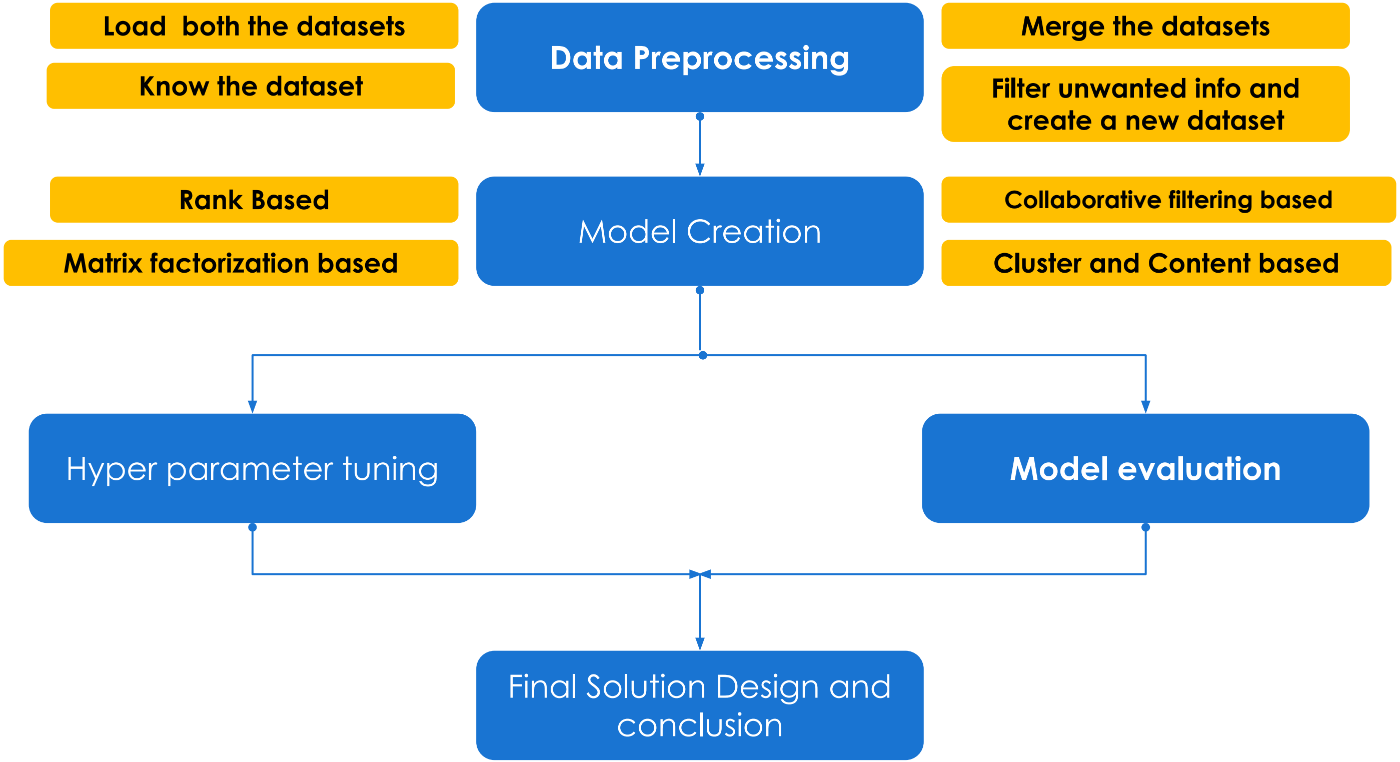
Song_data

Song_id	A unique id given to every song
title	Title of the song
Release	Name of the released album
Artist_name	Name of the artist
Year	Year of release

Count_data

user _id	A unique id given to the user
song_id	A unique id given to the song
play_count	Number of times the song was played

Solution Approach



Key Points

1) **Google colab -** It is advisable to work on google colab to complete this project. Google Colab runs in the cloud, which means users do not **need to install any software or configure any hardware to use it.** All the necessary computing resources are provided by Google's servers.Google Colab is **based on Jupyter Notebooks and is completely free to use.**

2) **Surprise library** - The surprise library is a Python library that is commonly used in recommendation systems and machine learning.

   a) The surprise library is easy to use and provides a high-level interface that abstracts away many of the details of building recommendation systems.

   b) The surprise library provides several evaluation metrics that can be used to evaluate the performance of recommendation algorithms. These metrics include accuracy metrics like Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and F1-score.

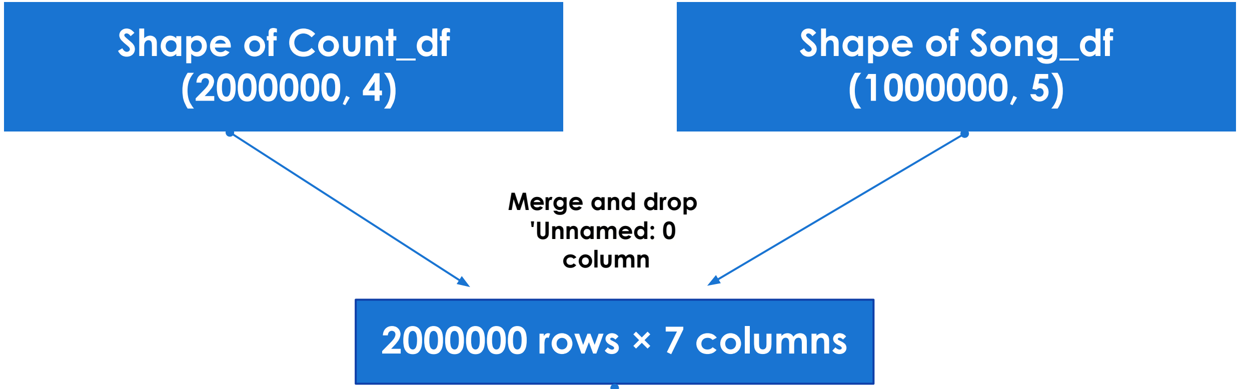
Installing surprise library

How to install surprise Library?

```
!pip install surprise
```

```
Collecting surprise
  Downloading surprise-0.1-py2.py3-none-any.whl (1.8 kB)
Collecting scikit-surprise
  Downloading scikit-surprise-1.1.1.tar.gz (11.8 MB)
     |████████████████████████████████| 11.8 MB 4.4 MB/s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.1.0)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.21.5)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.4.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.15.0)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.1-cp37-cp37m-linux_x86_64.whl size=1630183 sha256=d53b1630881315c1
  Stored in directory: /root/.cache/pip/wheels/76/44/74/b498c42be47b2406bd27994e16c5188e337c657025ab400c1c
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.1 surprise-0.1
```

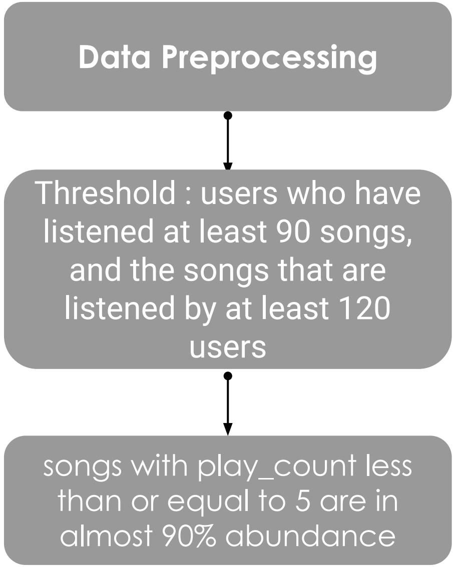
Cleaning the dataset



A dataset of size 2000000 rows x 7 columns can be quite large and may require a lot of computing resources to process. This can lead to long processing times and can make it difficult to train and evaluate your model efficiently.

In order to address this issue, it may be necessary to trim down your dataset to a more manageable size.

Probable steps to reduce the size



Observations and Insights

- Present the results of your recommendation system. This should include any metrics you used to **evaluate the performance of your model, such as accuracy or precision.** You may also want to include visualizations to help illustrate your findings

- Discuss any interesting observations or insights you gained from your work. This could include unexpected results or patterns you noticed in the data. Be sure to explain **why these findings are important and how they relate to the goals of your project.**

- **Summarize your findings and key takeaways from your work**. This will help ensure that your audience understands the significance of your work and its implications.

Pre-defined functions

The given function is used to calculate the precision@k, recall@k, RMSE, and F_1 score for a recommendation system model.

1. Create a dictionary "user_est_true" that maps the predictions made by the model to each user in the test dataset.
2. Loop through each user in the test dataset, and for each user, sort the predicted ratings by estimated value.
3. Calculate the number of relevant items, recommended items in top k, and relevant and recommended items in top k for each user.
4. Compute precision@k and recall@k for each user and store the values in dictionaries "precisions" and "recalls".
5. Compute the mean of all the predicted precisions and recalls.
6. Compute the RMSE score for the model on the test dataset using the "accuracy.rmse()" function.
7. Print the overall precision, recall, and F_1 score for the model on the test dataset using the computed mean precisions and recalls.
8. Return the RMSE score.

```python
# The function to calulate the RMSE, precision@k, recall@k, and F_1 score
def precision_recall_at_k(model, k = 30, threshold = 1.5):
    """Return precision and recall at k metrics for each user"""

    # First map the predictions to each user.
    user_est_true = defaultdict(list)

    # Making predictions on the test data
    predictions=model.test(testset)

    for uid, _, true_r, est, _ in predictions:
        user_est_true[uid].append((est, true_r))

    precisions = dict()
    recalls = dict()
    for uid, user_ratings in user_est_true.items():

        # Sort user ratings by estimated value
        user_ratings.sort(key = lambda x : x[0], reverse = True)
```

If you are attempting full code please modify the functions as per your code

Q&A