Music Recommendation System

This report builds a music recommendation system that combines Collaborative Filtering (CF) and Content Based Filtering (CBF). CF looks at the songs users have listened to, while CBF examines details like the artist, genre, and title of the songs. By using both methods, the system is able to give song suggestions that are both personal and varied based on the available dataset. This approach makes sure that the recommendations are tailored to the user's taste while also offering some new option. The next sections will explain how the system works and how the dataset supports the design choice.

1. **Installing Surprise**

First, a library called surprise was installed in Google Colab environment designed for building recommendation systems.

```
!pip install surprise
```

2. **Importing Libraries**

Then imports of all the necessary libraries like pandas for data manipulation, sklearn for some calculations and ipywidgets to help make interactive function was used for the recommendation algorithm.

```
import pandas as pd
from surprise import SVD, Dataset, Reader
from surprise.model_selection import train_test_split
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
import ipywidgets as widgets
from IPython.display import display
```

3. **Data Loading and Preprocessing**

The dataset is loaded into the environment which formats the data for processing.

Code

```
df_songsDB = pd.read_csv('song_dataset.csv')
df_songsDB
```

Result. Here is a snapshot of the dataset.

	user	song	play_count	title	release	artist_name	year
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1	The Cove	Thicker Than Water	Jack Johnson	0
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAPDEY12A81C210A9	1	Nothing from Nothing	To Die For	Billy Preston	1974
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2	Entre Dos Aguas	Flamenco Para Niños	Paco De Lucia	1976
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBFNSP12AF72A0E22	1	Under Cold Blue Stars	Under Cold Blue Stars	Josh Rouse	2002
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBFOVM12A58A7D494	1	Riot Radio (Soundtrack Version)	Nick & Norah's Infinite Playlist - Original Mo...	The Dead 60s	0
...	...	...	...	...	...	...	...
102622	21f4ac98aa1665bd42027ba12184a939ff435f59	SOKAKHH12AF72A5BAF	3	87	Hopeless Romantic	Bouncing Souls	1999
102623	21f4ac98aa1665bd42027ba12184a939ff435f59	SONPOXM12A8C1440C2	4	Space Olympics	Incredibad	The Lonely Island	2009
102624	21f4ac98aa1665bd42027ba12184a939ff435f59	SOPREHY12AB01815F9	8	I'm On A Boat	Incredibad	The Lonely Island / T-Pain	2009
102625	21f4ac98aa1665bd42027ba12184a939ff435f59	SOQXKUV12A6D4FB4C9	3	Amityville	The Marshall Mathers LP	Eminem / Bizarre	2000
102626	21f4ac98aa1665bd42027ba12184a939ff435f59	SOSJRJP12A6D4F826F	18	Master Of Puppets	Master Of Puppets	Metallica	1986

102627 rows × 7 columns

4. **Collaborative Filtering (to learn from user histories).**

Collaborative Filtering tracks the songs users listen to and how often they listen to them using the play_count to see which songs are favorites. When a song is played frequently, the system assumes it's a favorite of the user. It also identifies patterns in the listening habits of other users with similar tastes and suggests songs they might enjoy. This is achieved by using Singular Value Decomposition (SVD), a technique that simplifies the data, making it easier to detect these patterns. Here's how this method is implemented

```
reader = Reader(rating_scale=(1, df_songsDB['play_count'].max()))
surpriseData = Dataset.load_from_df(df_songsDB[['user', 'song', 'play_count']], reader)

trainset, testset = train_test_split(surpriseData, test_size=0.2, random_state=20)

algo_SVD = SVD()
algo_SVD.fit(trainset)
```

5. **Content Based Filtering is different from Collaborative Filtering.**

Instead of focusing on user behavior like Collaborative Filtering, Content Based Filtering looks at the features of the songs themselves, such as their title, genre, and artist. The frequency with which a song is played (play_count) is important because it helps the system understand which songs the user likely enjoys most. Songs with higher play counts are likely to be the user's favorites so this information is key to making good recommendations. To compare songs and find similar ones, the system uses a method called TF IDF (Term Frequency Inverse Document Frequency). This method converts song features into numbers which allows the system to measure how similar two songs are. This makes it possible to recommend songs with similar characteristics to the ones the user already enjoy. The implementation is as follows;

```
def content_score_calculator(uesr_songs, unlistened_songs):
    df_songsDB['combined_features'] = (
        df_songsDB['artist_name'] + " " +
        df_songsDB['release'] + " " +
        df_songsDB['title']
    )

    uesr_songs_features = df_songsDB[df_songsDB['title'].isin(uesr_songs)]['combined_features']
    unlistened_song_features = df_songsDB[df_songsDB['song'].isin(unlistened_songs)]['combined_features']

    tfidf = TfidfVectorizer()
    tfidf_matrix = tfidf.fit_transform(df_songsDB['combined_features'])

    selected_matrix = tfidf.transform(uesr_songs_features)
    unlistened_matrix = tfidf.transform(unlistened_song_features)
    similarity_scores = cosine_similarity(selected_matrix, unlistened_matrix)

    avg_similarity = similarity_scores.mean(axis=0)

    return dict(zip(unlistened_songs, avg_similarity))
```

6. In the hybrid model, collaborated filtering (CF) and content based filtering (CBF) are combined. When the alpha value is set to 0.5, the system does not favour one approach over the other unless there is a specific reason for doing so. Higher alpha values emphasise CF, which examines users' listening patterns while lower alpha values emphasise CBF which adds diversity to the suggestions. The algorithm guarantees that the recommendations are tailored by integrating the two methods, while still incorporating music with comparable characteristics. The hybrid model is implemented as follows;

```
def hybridRecommendationEngine(user_id):
    alpha=0.5
    user_songs=list(df_songsDB[df_songsDB['user']==user_id]['title'].unique())
    listened_songs = df_songsDB[df_songsDB['user'] == user_id]['song'].unique()
    all_songs = df_songsDB['song'].unique()
    unlistened_songs = set(all_songs) - set(listened_songs)

    cf_scores = collaborative_score_calculator(user_id, unlistened_songs)
    content_scores = content_score_calculator(user_songs, unlistened_songs)

    final_scores = {}
    for song_id in unlistened_songs:
        cf_score = cf_scores.get(song_id, 0)
        content_score = content_scores.get(song_id, 0)
        final_scores[song_id] = alpha * cf_score + (1 - alpha) * content_score

    scores = list(final_scores.values())
    min_score = min(scores) if scores else 0
    max_score = max(scores) if scores else 1

    if max_score > min_score:
        normalized_scores = {
            song_id: (score - min_score) / (max_score - min_score)
            for song_id, score in final_scores.items()
        }
    else:
        normalized_scores = {song_id: 0.5 for song_id in final_scores}

    sorted_songs = sorted(normalized_scores.items(), key=lambda x: x[1], reverse=True)
    recommended_song_ids = [song_id for song_id, _ in sorted_songs[:10]]

    recommended_songs = (
        pd.DataFrame(recommended_song_ids, columns=['song'])
        .merge(df_songsDB[['song', 'title', 'release', 'artist_name', 'year']].drop_duplicates(), on='song', how='lef
    )
    return recommended_songs
```

By adjusting alpha, the system can modify its approach for various situations. For instance, users with rich listening histories benefit from higher CF weights while newer users can rely more on CBF.

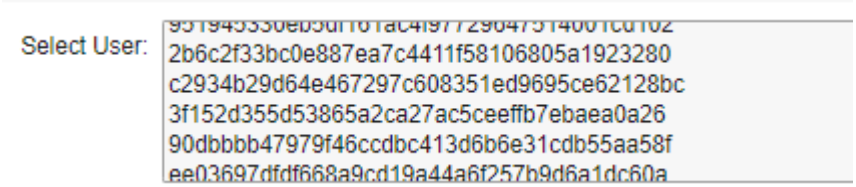
7. User Interface and recommendations.

The system is easy to use. To get personalised song recommendations, users click "Get Recommendations" after choosing their ID from a dropdown list. Because of the system's direct connection to the recommendation engine, it can quickly produce results based on user input. Users are assisted in making well informed decisions by the recommendations' clear table display, which includes the song, artist, and play count. To guarantee that the recommendations suit the user's preferences while providing some variation, the system may, for example, recommend more of Ed Sheeran's tracks or comparable ones by other artists if the user listens to his music frequently.

```
user_select = widgets.Select(
    options=df_songsDB.user.unique(),
    description='Select User:',
    disabled=False,
    layout=widgets.Layout(width='auto', height='auto'),
)
user_select.style.description_width = 'initial'
display(user_select)
```

Result of the User Selection

When the widget is executed, it displays a dropdown menu with all the unique user IDs from the dataset. A screenshot of the interface is shown below;



This allows the system to update recommendations instantly based on the user selected from the dropdown.

8. Get Recommendations Interface Step

The system uses a combination of Collaborative Filtering and Content Based Filtering to generate recommendations based on the user's listening history. When a user selects their ID, the system retrieves their previous interactions with music and processes them through a hybrid model. The alpha parameter within this model determines the weighting of each approach, allowing the system to balance personal preferences with diversity in the recommendations.

```
recommend_button = widgets.Button(
    description='Get Recommendations',
    disabled=False,
    button_style='success',
    tooltip='Click to get recommendations',
    icon='check'
)
display(recommend_button)

def on_recommend_button_click(b):
    user_id = user_select.value
    recommendations = hybridRecommendationEngine(user_id)
    print("Recommended Songs:")
    display(recommendations)

recommend_button.on_click(on_recommend_button_click)
```

Result

These scores are used by the backend to retrieve the most recommended songs which are then shown in the user interface. With the use of the alpha parameter, the suggestions can be made different for new listeners or more tailored for people with extensive listening histories. This smooth connection between the model and the interface makes the recommendation process easy and effective.

Get Recommendat...

Recommended Songs:

	song	title	release	artist_name	year
0	SOSJLSI12A8C13FB69	Your Rocky Spine	Penny Candy_ Vol. 2	Great Lake Swimmers	2007
1	SOMAKIT12A58A7E292	Bodies	Sinner	Drowning Pool	2001
2	SOQBUFQ12A6D4F7F4C	MIC (Speak Life Album Version)	Speak Life	Sev Statik	0
3	SOAUWYT12A81C206F1	Undo	Vespertine Live	Björk	2001
4	SOEKNHF12A3F1E9B8E	Times	Over And Underneath	Tenth Avenue North	2008
5	SOEGIYH12A6D4FC0E3	Horn Concerto No. 4 in E flat K495: II. Romanc...	Mozart - Eine kleine Nachtmusik	Barry Tuckwell/Academy of St Martin-in-the-Fie...	0
6	SOILVXC12AB017C10D	Batang-bata Ka Pa	Tala-Arawan	Sugar Free	0
7	SOSXLTC12AF72A7F54	Revelry	Only By The Night	Kings Of Leon	2008
8	SOZWBVE12AAF3B515C	Golden Mummy Golden Bird	Desperate Living	Horse The Band	2009
9	SOBONKR12A58A7A7E0	You're The One	If There Was A Way	Dwight Yoakam	1990

Recommended Songs:

	song	title	release	artist_name	year
0	SOSJLSI12A8C13FB69	Your Rocky Spine	Penny Candy_ Vol. 2	Great Lake Swimmers	2007
1	SOMAKIT12A58A7E292	Bodies	Sinner	Drowning Pool	2001
2	SOQBUFQ12A6D4F7F4C	MIC (Speak Life Album Version)	Speak Life	Sev Statik	0
3	SOAUWYT12A81C206F1	Undo	Vespertine Live	Björk	2001
4	SOEKNHF12A3F1E9B8E	Times	Over And Underneath	Tenth Avenue North	2008
5	SOEGIYH12A6D4FC0E3	Horn Concerto No. 4 in E flat K495: II. Romanc...	Mozart - Eine kleine Nachtmusik	Barry Tuckwell/Academy of St Martin-in-the-Fie...	0

Output. The Get Recommendations Interface turns the selected user ID into a meaningful output by generating and displaying personalized song suggestions. It is the final step in the interaction process, completing the recommendation cycle.

Why This Design Was Chosen

The design of the music recommendation system focuses on balancing personalization and variety in suggestions. Collaborative Filtering (CF) is used because it predicts user preferences based on their listening history, finding patterns in user song interactions. It works well with datasets that have enough user activity like this one. Singular Value Decomposition (SVD) was chosen for CF because it efficiently handles sparse data which is common in recommendation systems. To add more variety, Content Based Filtering (CBF) was included. This method looks at details like the song title, artist, genre, and how often it's played. It uses TF IDF to turn these details into numbers, making it easier to compare songs. The play count is important because it helps highlight the songs the user listens to most, making the suggestions more relevant.

The hybrid model combines CF and CBF using an alpha setting. This allows the system to strike a balance between personalized recommendations and new suggestions, ensuring the songs fit the user's taste while still offering variety. This setup works well because it's simple, flexible, and matches the needs of the dataset and project.

Possible Alternatives

The current system works well, but there are ways it could be improved. Using Neural Collaborative Filtering (NCF) could help understand more complex preferences especially when dealing with larger datasets. As for Content Based Filtering, adding audio features like tempo, pitch, and mood would improve how songs are compared not just by their text information. These methods could help address current limitations and improve the system's scalability and accuracy.

Conclusion.

The music recommendation system does a great job of suggesting songs that match what users like. The system combines two approaches, it considers the songs a user has listened to (Collaborative Filtering) as well as the song details like artist and genre (Content Based Filtering). This combination helps make sure the recommendations are not only tailored to the user's taste but also include a diverse range of option. The interface is simple, so users can easily get song suggestions with just a few clicks. Whether they're looking for something new or familiar or want to try new music, the system does a great balance between the two. Overall, this project shows how combining different approaches can create a system that is both useful and fun. It gives users exactly what they want, a songs they will enjoy, while also introducing them to new music they might not have discovered on their own.