Music Recommendation System

ASHWIN PAL
APPLIED DATA SCIENCE PROGRAM
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
2025-03-08

Introduction

Problem Statement & Objectives

- Problem

  Finding new music that matches personal taste is difficult for users.

- Objective

  Develop a recommendation system that suggests songs based on listening habits.

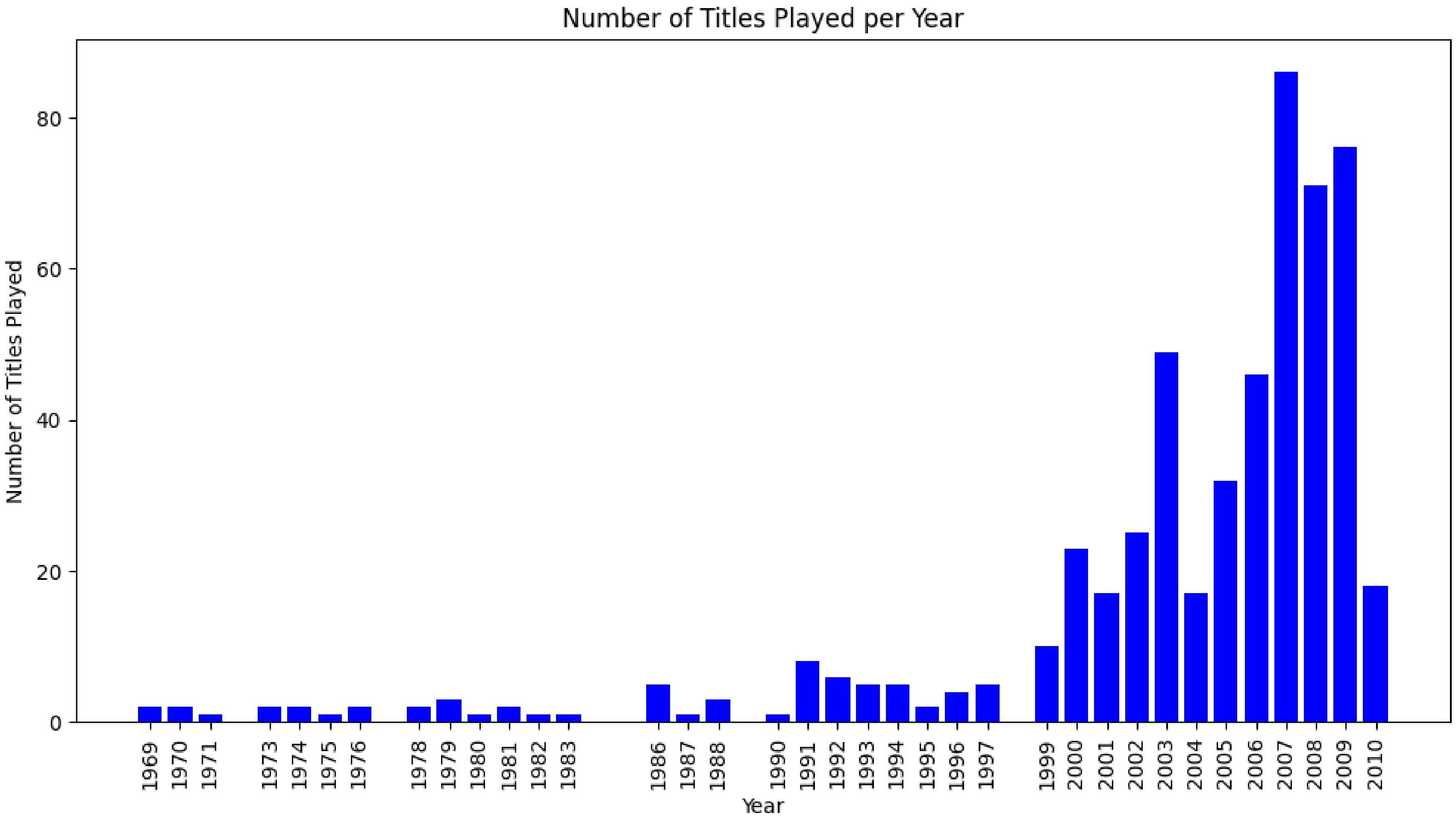
Dataset & Data collection

- 337 unique users
- 620 unique songs
- 247 artists

- 93.32% Sparsity

title	song_id	release	artist_name	year	user_id	play_count	text
Van Helsing Boombox	7900	Six Demon Bag	Man Man	2006	44689	1	Van Helsing Boombox Six Demon Bag Man Man
Sincerité Et Jalousie	617	Simple Et Funky	Alliance Ethnik	<NA>	34225	3	Sincerité Et Jalousie Simple Et Funky Alliance...
The Maestro	4954	Check Your Head	Beastie Boys	1992	27018	5	The Maestro Check Your Head Beastie Boys
Too Much Love	2557	LCD Soundsystem	LCD Soundsystem	2005	27018	4	Too Much Love LCD Soundsystem LCD Soundsystem
Porno Disaster	6482	Identification Parade	Octopus Project	2002	3139	2	Porno Disaster Identification Parade Octopus P...

Exploratory Data Analysis (EDA)



Modeling Techniques

Rank	Model	RMSE	Precision	Recall	F1-Score	Remarks
🥇 1	**SVD**	0.9948	0.428	0.650	0.516	Best RMSE (Most Accurate)
🥈 2	**Sim User-User Optimized**	1.0175	0.445	0.647	0.527	Best Precision & F1-Score
🥉 3	**Sim User-User**	1.0758	0.403	0.707	0.513	Best Recall (Widest Coverage)
4	**SVD Optimized**	1.0023	0.406	0.642	0.497	Slightly Lower RMSE Than User-User
5	**Sim Item-Item Optimized**	1.0171	0.346	0.551	0.425	Good Precision but Lower Recall
6	**Sim Item-Item**	1.0244	0.315	0.575	0.407	Lower Performance Than Optimized
7	**Clust Baseline**	1.0376	0.399	0.590	0.476	Performs Worse Than Other Models
8	**Clust Tuned**	1.0374	0.398	0.589	0.475	Minimal Improvement Over Baseline

Proposed Model Solution

- **Hybrid Recommendation System (SVD & User-User Based)**

Rank	Model	RMSE	Precision	Recall	F1-Score	Remarks
🥇 1	**SVD Hybrid**	0.9743	0.424	0.613	0.501	Best RMSE (Most Accurate Hybrid Model)
🥈 2	**SVD**	0.9948	0.428	0.650	0.516	Strong RMSE, Best Recall
🥉 3	Sim User-User Optimized	1.0175	0.445	0.647	0.527	Best Precision & F1-Score
4	**SVD Optimized**	1.0023	0.406	0.642	0.497	Slightly Lower RMSE Than User-User
5	Sim User-User	1.0758	0.403	0.707	0.513	Best Recall (Widest Coverage)
6	Sim Item-Item Optimized	1.0171	0.346	0.551	0.425	Good Precision but Lower Recall
7	Sim Item-Item	1.0244	0.315	0.575	0.407	Lower Performance Than Optimized
8	Clust Baseline	1.0376	0.399	0.590	0.476	Performs Worse Than Other Models
9	Clust Tuned	1.0374	0.398	0.589	0.475	Minimal Improvement Over Baseline

Challenges

- **Computational Costs**
- **Scalability:**
- **Data Bias**
- **Privacy & Security**
- **Sustainability**

Summary:

- Built a Hybrid Recommendation System using SVD & User-Based Filtering.
- Balanced accuracy & diversity, improving recommendations.
- Enhanced user engagement, making the system scalable for streaming services.
- Future improvements: Deep learning & real-time updates.

Thank You ☺

Appendix

	Count	Percentage
title		
Use Somebody	1602	1.16

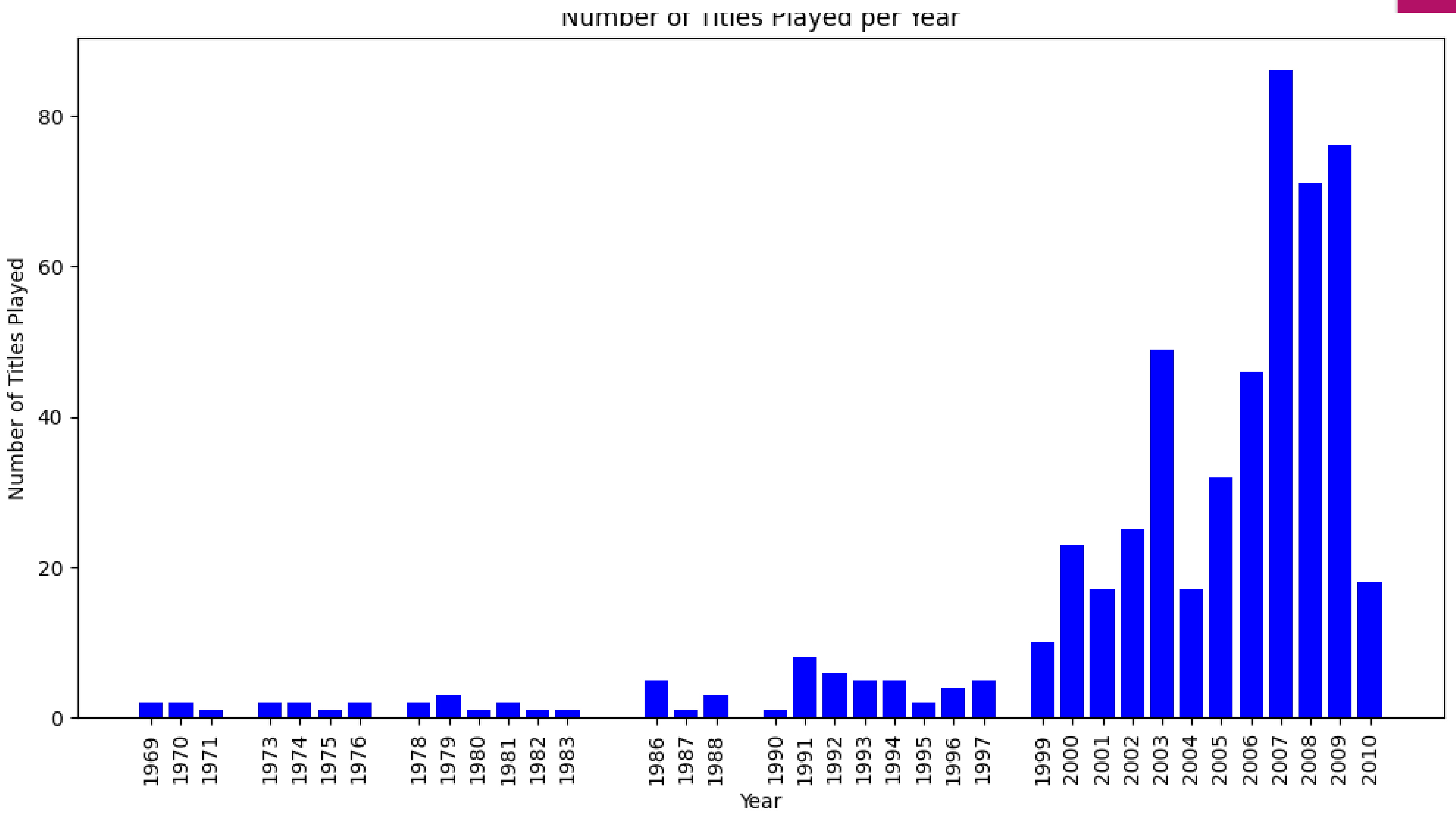
```
-----------------------------
Column: release
```

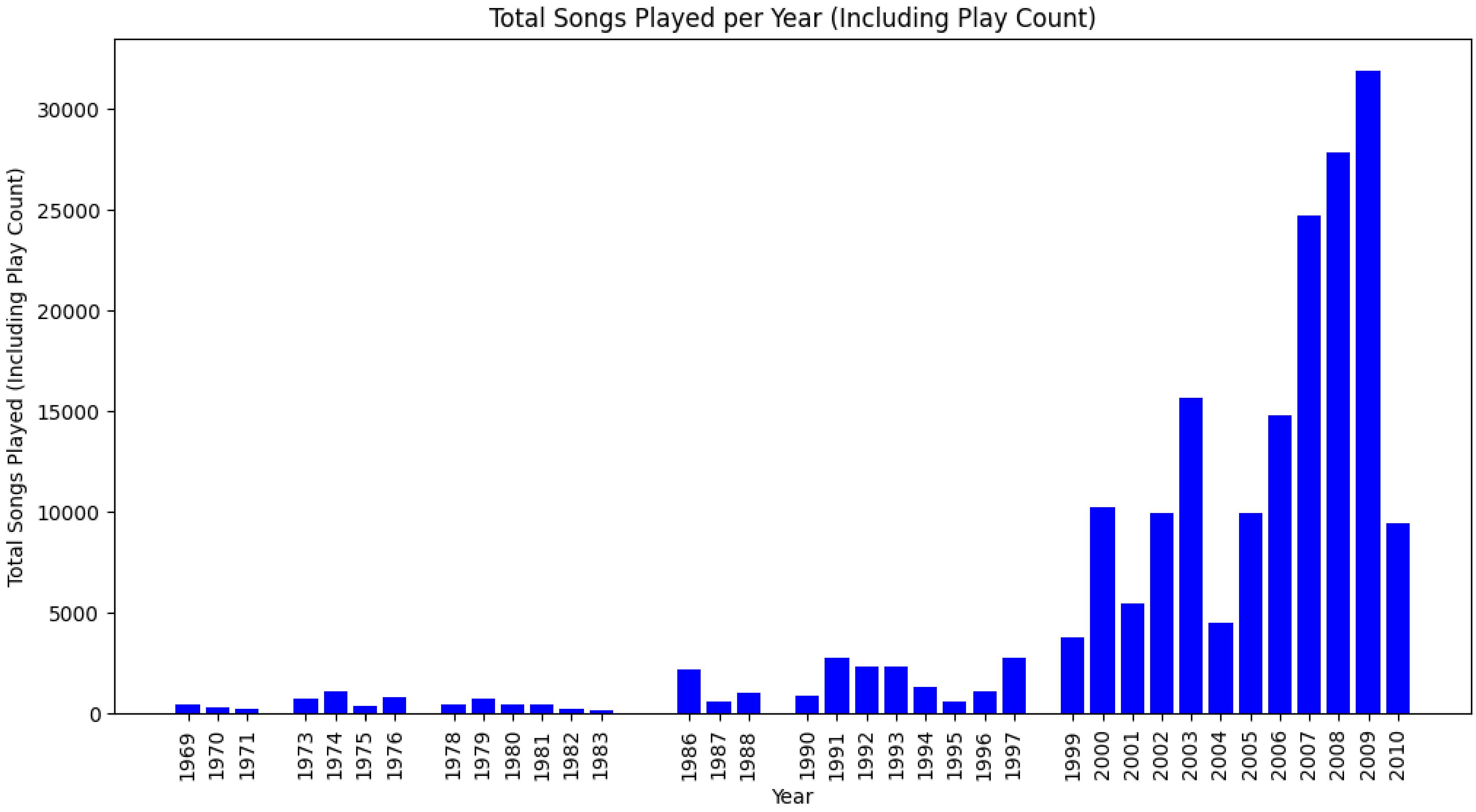
	Count	Percentage
release		
My Worlds	1967	1.42

```
-----------------------------
Column: artist_name
```

	Count	Percentage
artist_name		
Coldplay	6527	4.72

```
Number of unique values in song_id: 620
Number of unique values in title: 629
Number of unique values in release: 453
Number of unique values in artist_name: 247
Number of unique values in year: 37
Number of unique values in user_id: 3337
Number of unique values in play_count: 5
```


Number of Titles Played per Year
Number of Titles Played
Year
0
20
40
60
80
1969
1970
1971
1973
1974
1975
1976
1978
1979
1980
1981
1982
1983
1986
1987
1988
1990
1991
1992
1993
1994
1995
1996
1997
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010



Total Songs Played per Year (Including Play Count)
Total Songs Played (Including Play Count)
0
5000
10000
15000
20000
25000
30000
1969
1970
1971
1973
1974
1975
1976
1978
1979
1980
1981
1982
1983
1986
1987
1988
1990
1991
1992
1993
1994
1995
1996
1997
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
Year

Solution Code 1

```
# Build baseline model using svd
hybrid_svd = SVD(n_factors=150, biased=True, random_state=1)
  # n_factor: latent factors (user & item features extracted from interactions
  # biased: Determines whether to include user and item biases in the prediction
    # True: model learns biases for users and items to improve accuracy

# Training the algorithm on the training dataset
hybrid_svd.fit(trainset)
```

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x79264c874c10>
```

```
# Let us compute precision@k, recall@k, and f_1 score with k = 10
precision_recall_at_k(hybrid_svd)
```

```
RMSE: 0.9743
Precision:  0.424
Recall:  0.613
F_1 score:  0.501
```

Solution Code 2

Compute User-Based Similarity

```
# Aggregate duplicate entries by summing play counts
df_cleaned = df.groupby(['user_id', 'song_id'])['play_count'].sum().reset_index()

# Create user-song matrix
user_song_matrix = df_cleaned.pivot(index='user_id', columns='song_id', values='play_count').fillna(0)

# Compute cosine similarity between users
user_similarity = cosine_similarity(user_song_matrix)

# Convert similarity matrix into DataFrame
user_sim_df = pd.DataFrame(user_similarity, index=user_song_matrix.index, columns=user_song_matrix.index)
```

Solution Code 3

Hybrid Recommendation Function

```
def hybrid_recommend(user_id, song_id):
    # Get SVD predicted rating
    pred_rating = hybrid_svd.predict(user_id, song_id).est

    # Get user-based similarity score
    if user_id in user_sim_df.index:
        similar_users = user_sim_df[user_id].sort_values(ascending=False)[1:6]  # Top 5 similar users
        user_based_score = df_cleaned[
            (df_cleaned['user_id'].isin(similar_users.index)) &
            (df_cleaned['song_id'] == song_id)
        ]['play_count'].mean()
    else:
        user_based_score = 0  # Default score if no similar users

    # Hybrid Score: Weighted sum of SVD prediction and user-based score
    hybrid_score = (0.7 * pred_rating) + (0.3 * (user_based_score if not np.isnan(user_based_score) else 0))

    return hybrid_score
```

Solution Code Predictions

Make Predictions for Specific Users & Songs

```
# Making prediction for user (user_id=6958) to song (song_id=1671), r_ui=2
hybrid_svd.predict(6958, 1671, r_ui=2, verbose=True)
```

```
user: 6958       item: 1671       r_ui = 2.00   est = 1.31   {'was_impossible': False}
Prediction(uid=6958, iid=1671, r_ui=2, est=1.3136149649009117, details={'was_impossible': False})
```

Predict Play Count for a User Who Has NOT Listened to a Song

```
# Predict play count for a user who has NOT listened to a song
hybrid_svd.predict(37684, 7737, verbose=True)
```

```
user: 37684      item: 7737       r_ui = None   est = 2.16   {'was_impossible': False}
Prediction(uid=37684, iid=7737, r_ui=None, est=2.161421864352362, details={'was_impossible': False})
```

Rank	Model	RMSE	Precision	Recall	F1-Score	Remarks
🥇 1	**SVD Hybrid**	0.9743	0.424	0.613	0.501	Best RMSE (Most Accurate Hybrid Model)
🥈 2	**SVD**	0.9948	0.428	0.650	0.516	Strong RMSE, Best Recall
🥉 3	Sim User-User Optimized	1.0175	0.445	0.647	0.527	Best Precision & F1-Score
4	**SVD Optimized**	1.0023	0.406	0.642	0.497	Slightly Lower RMSE Than User-User
5	Sim User-User	1.0758	0.403	0.707	0.513	Best Recall (Widest Coverage)
6	Sim Item-Item Optimized	1.0171	0.346	0.551	0.425	Good Precision but Lower Recall
7	Sim Item-Item	1.0244	0.315	0.575	0.407	Lower Performance Than Optimized
8	Clust Baseline	1.0376	0.399	0.590	0.476	Performs Worse Than Other Models
9	Clust Tuned	1.0374	0.398	0.589	0.475	Minimal Improvement Over Baseline