```
# Source code for AI-driven movie matchmaking recommendation systems
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer

# Sample movie dataset
movies = pd.DataFrame({ 'MovieID': [1, 2, 3, 4],'Title': ['Inception', 'Titanic',
'Avatar', 'The Matrix'], 'Genres': ['Sci-Fi Thriller', 'Romance Drama', 'Sci-Fi
Action', 'Sci-Fi Action']})

# Sample user rating dataset
ratings = pd.DataFrame({ 'UserID': [1, 1, 2, 2, 3], 'MovieID': [1, 2, 2, 3, 4],
'Rating': [5, 3, 4, 5, 4]})

# Step 1: Content-Based Filtering using Genre
tfidf = TfidfVectorizer()
genre_matrix = tfidf.fit_transform(movies['Genres'])
genre_sim = cosine_similarity(genre_matrix)

# Step 2: Build user-movie matrix
user_movie_matrix = ratings.pivot_table(index='UserID',
columns='MovieID',values='Rating').fillna(0)

# Step 3: Recommend for a specific user
def recommend_movies(user_id):
    seen_movies = ratings[ratings['UserID'] == user_id]['MovieID'].tolist()
    scores = pd.Series(dtype='float64')
    for movie_id in seen_movies:
        idx = movies[movies['MovieID'] == movie_id].index[0]
        sim_scores = list(enumerate(genre_sim[idx]))

        for i, score in sim_scores:
            if movies.loc[i, 'MovieID'] not in seen_movies:
                scores[movies.loc[i, 'Title']] = scores.get(movies.loc[i, 'Title'], 0) +
score

    recommendations = scores.sort_values(ascending=False)
    return recommendations.head(3)

# Output for user 1
print("Top Recommendations for User 1:")
print(recommend_movies(1))
```