INTERN:- Bhagyashri Sharad Pisal

*Intern ID:- CT4MOTH

*Domain:- Machine Learning

*Duration:-January 25, 2025, to May 25, 2025

*Company:- CODETECH IT SOLUTIONS

*Mentor:- Neela Santhosh Kumar

TASK FOUR:RECOMMENDATION SYSTEM

BUILD A RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING OR MATRIX FACTORIZATION

DELIVERABLE: A NOTEBOOK OR APP SHOWCASING RECOMMENDATION RESULTS AND EVALUATION METRICS.

```
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import mean_squared_error
from math import sqrt


# Sample dataset: User-Item Ratings
ratings_data = {
    'User': ['A', 'A', 'B', 'B', 'C', 'C', 'D', 'D'],
    'Item': ['Item1', 'Item2', 'Item1', 'Item3', 'Item2', 'Item3', 'Item1', 'Item3'],
    'Rating': [15, 23, 14, 74, 13, 19,14, 12]
}
ratings_df = pd.DataFrame(ratings_data)


# Pivot the data to create a User-Item matrix
user_item_matrix = ratings_df.pivot(index='User', columns='Item', values='Rating').fillna(0)
print("User-Item Matrix:")
print(user_item_matrix)
```

```
User-Item Matrix:
Item  Item1  Item2  Item3
User
A      15.0   23.0    0.0
B      14.0    0.0   74.0
C       0.0   13.0   19.0
D      14.0    0.0   12.0
```

```
# Calculate cosine similarity between users
user_similarity = cosine_similarity(user_item_matrix)
user_similarity_df = pd.DataFrame(user_similarity, index=user_item_matrix.index, columns=user_item_matrix.index)
print("\nUser Similarity Matrix:")
print(user_similarity_df)
```

```
User Similarity Matrix:
User         A         B         C         D
User
A     1.000000  0.101547  0.472985  0.414757
B     0.101547  1.000000  0.810922  0.780588
C     0.472985  0.810922  1.000000  0.537103
D     0.414757  0.780588  0.537103  1.000000
```

```
# Function to predict ratings based on user similarity
def predict_ratings(user_item_matrix, similarity_matrix):
    predictions = np.zeros(user_item_matrix.shape)
    for i in range(user_item_matrix.shape[0]):  # For each user
        for j in range(user_item_matrix.shape[1]):  # For each item
            # Only predict if the user hasn't rated the item
            if user_item_matrix.iloc[i, j] == 0:
                # Weighted sum of ratings by similar users
                numerator = np.dot(similarity_matrix[i], user_item_matrix.iloc[:, j])
                denominator = np.sum(similarity_matrix[i])
                predictions[i, j] = numerator / denominator if denominator != 0 else 0
            else:
                predictions[i, j] = user_item_matrix.iloc[i, j]
    return predictions
```

Generate print hello world using rot13 Close

```
# Predict ratings
predicted_ratings = predict_ratings(user_item_matrix, user_similarity)
predicted_ratings_df = pd.DataFrame(predicted_ratings, index=user_item_matrix.index, columns=user_item_matrix.columns)
print("\nPredicted Ratings:")
print(predicted_ratings_df)
```

```
Predicted Ratings:
Item      Item1      Item2      Item3
User
A     15.000000  23.000000  10.796951
B     14.000000   4.781765  74.000000
C      9.204904  13.000000  19.000000
D     14.000000   6.046504  12.000000
```

```
# Evaluate the system using Root Mean Square Error (RMSE)
def calculate_rmse(actual, predicted):
    # Flatten matrices and calculate RMSE
    actual_flat = actual.values.flatten()
    predicted_flat = predicted.values.flatten()
    mse = mean_squared_error(actual_flat, predicted_flat)
    return sqrt(mse)


# Example evaluation (Assuming predicted_ratings is compared to original ratings_df)
evaluation_matrix = user_item_matrix.copy()
evaluation_rmse = calculate_rmse(evaluation_matrix, predicted_ratings_df)
print(f"\nRecommendation System RMSE: {evaluation_rmse:.2f}")
```

```
Recommendation System RMSE: 4.66
```

```
# Recommend top-N items for each user
def recommend_items(predicted_ratings_df, n=2):
    recommendations = {}
```

```
    for user in predicted_ratings_df.index:
        sorted_items = predicted_ratings_df.loc[user].sort_values(ascending=False)
        recommendations[user] = sorted_items.head(n).index.tolist()
    return recommendations
```

Generate randomly select 5 items from a list Close

```
# Get top-2 recommendations for each user
recommendations = recommend_items(predicted_ratings_df, n=2)
print("\nTop-2 Recommendations for Each User:")
for user, items in recommendations.items():
    print(f"User {user}: {items}")
```

```
Top-2 Recommendations for Each User:
User A: ['Item2', 'Item1']
User B: ['Item3', 'Item1']
User C: ['Item3', 'Item2']
User D: ['Item1', 'Item3']
```