Experiment No. 6

Title - Implement a program for Recommendation System

Theory -

A recommendation system is a type of information filtering system that predicts and suggests items (products, services, content) to users based on their preferences or behaviours. These systems aim to enhance user experience by providing personalized and relevant recommendations, often leading to increased engagement and satisfaction. Recommendation systems find applications in various domains, such as e-commerce, streaming services, social media, and more.

Types of Recommendation Systems:

1. **Collaborative Filtering:**

- Collaborative filtering methods make predictions by leveraging the preferences and behaviours of other users.
- User-Based Collaborative Filtering: Recommends items to a user based on the preferences of users similar to them.
- Item-Based Collaborative Filtering: Recommends items similar to those liked by the user.

2. **Content-Based Filtering:**

- Content-based methods recommend items based on their features and the user's preferences.
- Item features, such as keywords or genres, are used to build user profiles and make recommendations.

3. **Hybrid Methods:**

- Hybrid recommendation systems combine collaborative and content-based filtering to leverage the strengths of both approaches.
- Techniques include weighted hybrid, switching hybrid, and feature combination hybrid.

4. **Matrix Factorization:**

- Matrix factorization techniques, such as Singular Value Decomposition (SVD), decompose the user-item interaction matrix into latent factors.
- Latent factors represent hidden patterns or features in the data, capturing user and item characteristics.

Singular Value Decomposition (SVD):

Overview:

- SVD is a linear algebra technique that decomposes a matrix into three other matrices, representing the singular vectors and singular values
- In the context of recommendation systems, the matrix often represents user-item interactions.

1. **Decomposition:**

- Given a matrix M, SVD decomposes it into three matrices: U , Σ , and V^T.
- U contains the left singular vectors, Σ is a diagonal matrix of singular values, and V^T contains the transposed right singular vectors.
- $M = U\Sigma V^T$

2. **Dimensionality Reduction:**

- The singular values in Σ indicate the importance of the corresponding singular vectors.
- By keeping only the top k singular values and vectors, the dimensionality of the matrix is reduced, capturing the most significant patterns.

$$M \approx U_k \Sigma_k V_k^T$$

3. **Recommendations:**

- The reduced matrices U_k, Σ_k, and V_k^T are used to make predictions.
- For user i and item j, the predicted rating $\hat{r}_{ij}$ can be calculated as the dot product of the corresponding vectors.

$$\hat{r}_{ij} = (U_k)_i \cdot (\Sigma_k)_i \cdot (V_k)_j^T$$

4. **Application to Recommendation Systems:**

- SVD helps capture latent factors in user-item interactions, enabling personalized recommendations.
- The learned representations in the reduced matrices can handle missing values and improve the model's generalization to new data.

Challenges and Considerations:

1. **Data Sparsity:**

Recommendation systems often deal with sparse user-item interaction matrices, where most entries are missing. Techniques like matrix factorization help address the sparsity issue.

2. **Scalability:**

As datasets grow, computational efficiency becomes crucial. Approximate methods and optimizations are often employed to handle large-scale matrices.

3. **Cold Start Problem:**

For new users or items with limited interactions, recommendation systems face challenges in making accurate predictions. Hybrid methods and content-based approaches can be used to mitigate the cold start problem.

4. **Evaluation Metrics:**

Common metrics for evaluating recommendation systems include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Precision, Recall, and F1-score.

In practice, Singular Value Decomposition is just one approach among many for building recommendation systems. It is particularly effective in collaborative filtering scenarios where user-item interactions are available. Implementing and fine-tuning recommendation systems often involve a combination of techniques tailored to the specific characteristics of the data and the application domain.

Implementation :

```
from surprise import Dataset, SVD
from surprise.model_selection import train_test_split
from surprise import accuracy
# Load the MovieLens dataset
data = Dataset.load_builtin('ml-100k')

# Define the algorithm (Singular Value Decomposition - SVD)
algo = SVD()

# Split the data into training and testing sets
trainset, testset = train_test_split(data, test_size=0.25)

# Train the algorithm on the training set
algo.fit(trainset)
# Make predictions on the test set
predictions = algo.test(testset)

# Evaluate the accuracy of the model
accuracy.rmse(predictions)

# Get recommendations for a specific user (user ID: 196)
user_id = str(196)
user_ratings = [(trainset.to_raw_iid(item), algo.predict(user_id, item).est) for item in
trainset.all_items() if item not in trainset.ur[trainset.to_inner_uid(user_id)]]
sorted_ratings = sorted(user_ratings, key=lambda x: x[1], reverse=True)

print(f"Top 5 recommended movies for user {user_id}:")
for movie_id, predicted_rating in sorted_ratings[:5]:
   print(f"Movie ID: {movie_id}, Predicted Rating: {predicted_rating:.2f}")
```

Output

```
RMSE: 0.9430
Top 5 recommended movies for user 196:
Movie ID: 773, Predicted Rating: 3.47
Movie ID: 1028, Predicted Rating: 3.47
Movie ID: 1078, Predicted Rating: 3.47
Movie ID: 421, Predicted Rating: 3.47
Movie ID: 288, Predicted Rating: 3.47
```