<AIE425 Intelligent Recommender Systems, Fall Semester 24/25>

<Assignment #2: Significance Weighting-based Neighborhood CF Filters>

<221101235, Marina Reda Abdullah Mekhael>

Table of Contents:

Introduction:

This assignment explores the impact of **weighting schemes** on the performance of **collaborative filtering (CF) recommender systems**. Collaborative filtering is a widely used technique in recommender systems that predicts user preferences based on past behavior and similar users or items. The assignment is divided into **two parts**:

1. **Part 1**: Demonstrate the significance of weighting schemes in the context of **user-based collaborative filtering (CF)**.
2. **Part 2**: Demonstrate the significance of weighting schemes in the context of **item-based collaborative filtering (CF)**.

The data collected in the first assignment was used which consists of **sports equipment sold on Amazon**. The data was collected using **web scraping** techniques, But we pooled the data again to increase the number of items to fit this important. followed by preprocessing steps (cleaned the data and changed the id of the users and the elements and arranged it) for easy reading. and adjusting ratings to a **1-to-5 scale** for uniformity.

To evaluate the performance of the recommender systems:

- Three active users (U1, U2, and U3) were selected with **2, 3, and 5 missing ratings**, respectively.
- Two target items (I1 and I2) were chosen with **4% and 10% missing ratings**, respectively.

This setup enables a focused analysis of how different weighting schemes impact the prediction accuracy and overall performance of both **user-based** and **item-based collaborative filtering systems.** The following report provides a detailed methodology, implementation process, results, and a discussion on the findings, highlighting the role of weighting schemes in improving recommendation quality.

Methodology:

This study evaluates the impact of weighting schemes in both user-based and item-based collaborative filtering (CF) systems. The methodology involves data preparation, similarity calculation, and evaluation of the recommendation performance, as described below:

Data Collection and Preprocessing

The dataset comprises sports equipment reviews collected from Amazon through web scraping. To enhance the scope of this analysis, the dataset was expanded and preprocessed as follows:

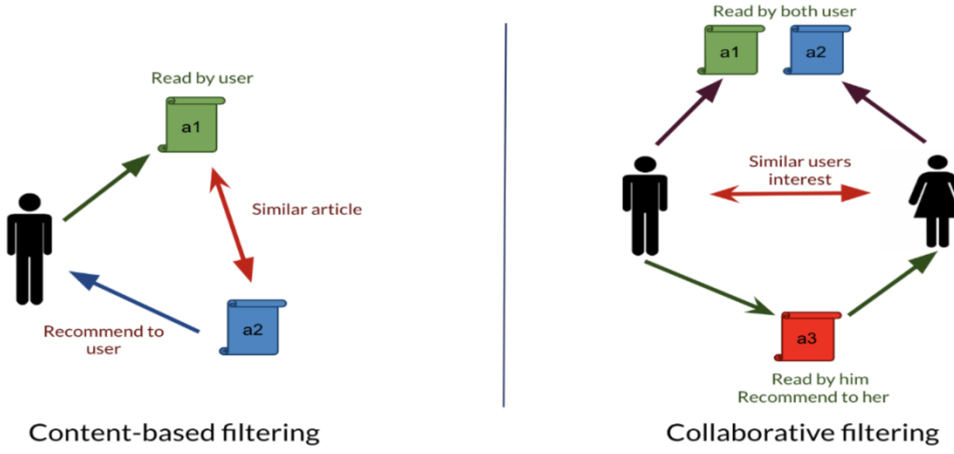
- **Cleaning**: Duplicate and inconsistent entries were removed to ensure data integrity.
- **User and Item ID Standardization**: User and item IDs were edited and arranged in a consistent format.
- **Rating Adjustment**: Ratings were normalized to a 1-to-5 scale to maintain uniformity across user preferences.
- **Target Selection**: Three active users (U1, U2, and U3) with 2, 3, and 5 missing ratings, respectively, and two target items (I1 and I2) with 4% and 10% missing ratings were chosen for focused analysis.

Collaborative Filtering Approaches

Two distinct approaches were implemented to analyze the impact of weighting schemes on CF systems:

- **User-Based Collaborative Filtering (User-CF)**: Recommendations are generated by identifying similar users based on their rating patterns and predicting the active user's missing ratings using their neighbors' preferences.
- **Item-Based Collaborative Filtering (Item-CF)**: Recommendations are generated by identifying similar items based on user rating patterns and

predicting missing ratings for target items based on their similarity to other items.



Similarity Measures

The similarity between users and items was calculated using two popular methods:

- **Cosine Similarity:** Measures the cosine of the angle between two rating vectors, focusing on the relative relationship between ratings.

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \cdot \sqrt{\sum_{i=1}^{n} v_i^2}}$$

- **Pearson Correlation Coefficient (PCC):** Captures the linear correlation between two rating vectors by adjusting for mean-centered biases.

$$\text{PCC} = \frac{\sum_{i=1}^{n} (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum_{i=1}^{n} (u_i - \bar{u})^2} \cdot \sqrt{\sum_{i=1}^{n} (v_i - \bar{v})^2}}$$

Significance Weighting and Discounting

Significance weighting was incorporated to adjust the similarity calculations based on the number of co-rated items or users. This helps reduce the influence of sparsely overlapping data:

- **No Discount**: Baseline similarity calculation without additional weighting.
- **Discount Applied**: Similarities were discounted by reducing the weights of comparisons with fewer co-ratings.

For item-based CF, discounting helped refine the recommendations by emphasizing only the most relevant item relationships.

Adjust the similarity scores by computing a Discount Factor (DF) based on the formula:

$$DF_{u,v} = \frac{\min(\beta, \text{common\_items})}{\beta}$$

Implementation Details

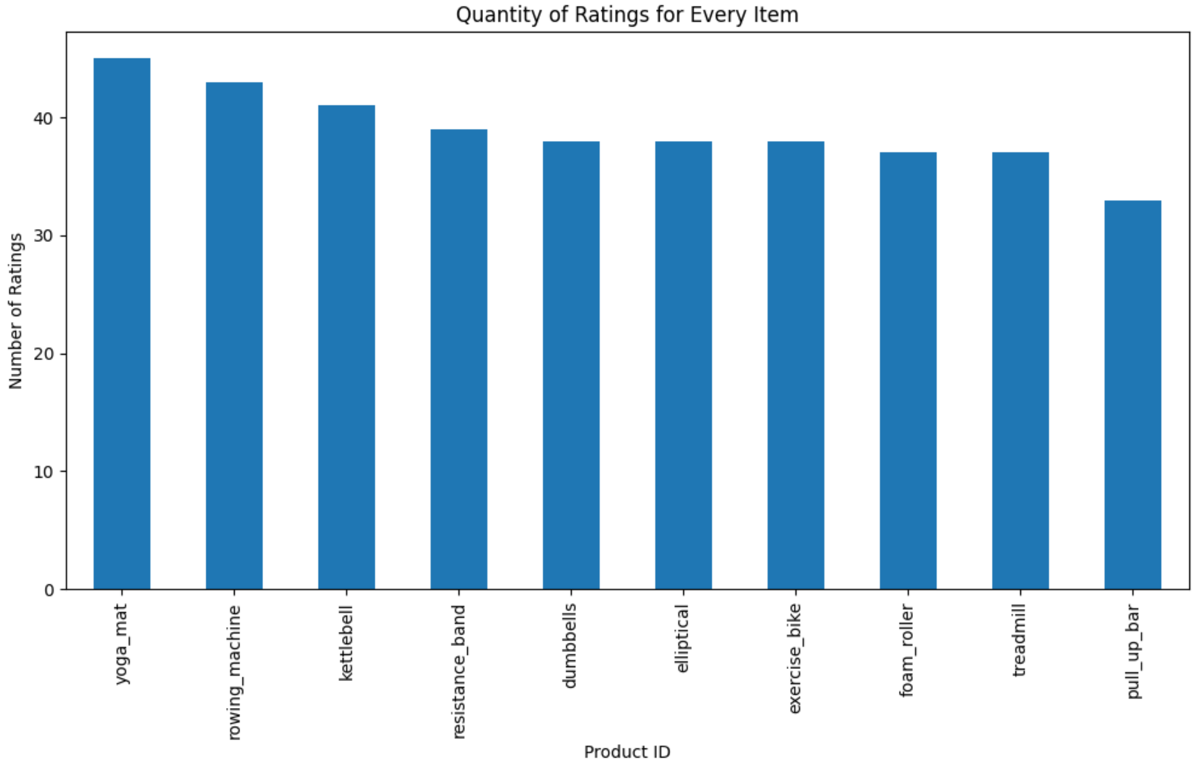
- The experiment was implemented in Python using libraries such as NumPy and Pandas for data manipulation and Scikit-learn for similarity calculations.
- Results, including co-ratings and recommendation rankings, were logged in assignment2_results.json for detailed analysis.

Outcomes of Section 3.1:


```
 "tnu": 50,
  "tni": 10,
  "ratings_per_product": {
     "yoga_mat": 45,
     "rowing_machine": 43,
     "kettlebell": 41,
     "resistance_band": 39,
     "dumbbells": 38,
     "elliptical": 38,
     "exercise_bike": 38,
     "foam_roller": 37,
     "treadmill": 37,
     "pull_up_bar": 33
  },
"thresholds": {
     "User7": 10,
     "User10": 9,
     "User39": 7
```

```
},
"active_users": [
    "User7",
    "User10",
    "User39"
],
"target_items": [
    "rowing_machine",
    "pull_up_bar"
]
```


To see "co_ratings" look at assignment2_results.json on GitHub Because it is big and will take more than 20 pages here



Summary of the Comparison of Part 1 and Part 2:

Part 1: User-Based Collaborative Filtering (CF):

- **Discount Impact**:
    - The inclusion of **discounting** did not significantly affect the **user similarity results** in this case. Whether or not discounting was applied, the rankings of **Top 20% Users** remained unchanged across **Cosine Similarity** and **Pearson Correlation Coefficient (PCC)** measures.
- **Cosine similarity considering the bias adjustment effect of mean-centering and PCC with Mean-Centering:**
    - Both **Cosine Similarity with the bias adjustment of MC** and **PCC** produce very similar results, mainly because they both adjust for **individual user biases** using **mean-centering**. This adjustment emphasizes the **relative preferences** between users for items, rather than their absolute ratings. As a result, these methods produced nearly identical rankings of users, further confirming that **bias adjustment** is the key factor in similarity calculations.

Compare and comment on the results of Case study 1.1, 1.2, and 1.3.

```
# 1.1.7: Compare results (similarity matrices)
for user in active_users:
    print(f"User: {user}")
    print("Top 20% Users (No Bias):", top_20_users[user].index.tolist())
    print("Top 20% Users (With Discount):", top_20_users_discounted[user].index.tolist())
    print()

#1.2.7: Compare the Results of Step 1.2.2 with Step 1.2.5
for user in active_users:
    print(f"User: {user}")
    print("Top 20% Users (Cosine Similarity):", top_20_users_centered[user].index.tolist())
    print("Top 20% Users (Discounted Similarity):", top_20_users_discounted_centered[user].index.tolist
    print()

#1.3.7: Compare Results of 1.3.2 and 1.3.5
for user in active_users:
    print(f"User: {user}")
    print("Top 20% Users (PCC):", top_20_users_pcc[user].index.tolist())
    print("Top 20% Users (Discounted PCC):", top_20_users_discounted_pcc[user].index.tolist())
    print()
```

```
User: User7
Top 20% Users (No Bias): ['User27', 'User30', 'User16', 'User50', 'User33', 'User28', 'User38', 'User24', 'User2']
Top 20% Users (With Discount): ['User27', 'User30', 'User16', 'User50', 'User33', 'User28', 'User38', 'User24', 'User2']

User: User10
Top 20% Users (No Bias): ['User30', 'User27', 'User32', 'User41', 'User3', 'User2', 'User40', 'User38', 'User7']
Top 20% Users (With Discount): ['User30', 'User27', 'User32', 'User41', 'User3', 'User2', 'User40', 'User38', 'User7']

User: User39
Top 20% Users (No Bias): ['User37', 'User46', 'User19', 'User6', 'User13', 'User41', 'User40', 'User23', 'User24']
Top 20% Users (With Discount): ['User37', 'User46', 'User19', 'User6', 'User13', 'User41', 'User40', 'User23', 'User24']

User: User7
Top 20% Users (Cosine Similarity): ['User27', 'User30', 'User16', 'User50', 'User33', 'User6', 'User17', 'User10', 'User37']
Top 20% Users (Discounted Similarity): ['User27', 'User30', 'User16', 'User50', 'User33', 'User6', 'User17', 'User10', 'User37']

User: User10
Top 20% Users (Cosine Similarity): ['User30', 'User27', 'User32', 'User3', 'User41', 'User7', 'User44', 'User9', 'User42']
Top 20% Users (Discounted Similarity): ['User30', 'User27', 'User32', 'User3', 'User41', 'User7', 'User44', 'User9', 'User42']

User: User39
Top 20% Users (Cosine Similarity): ['User37', 'User46', 'User19', 'User13', 'User6', 'User41', 'User40', 'User24', 'User23']
Top 20% Users (Discounted Similarity): ['User37', 'User46', 'User19', 'User13', 'User6', 'User41', 'User40', 'User24', 'User23']

User: User7
Top 20% Users (PCC): ['User27', 'User30', 'User16', 'User50', 'User33', 'User6', 'User17', 'User10', 'User37']
Top 20% Users (Discounted PCC): ['User27', 'User30', 'User16', 'User50', 'User33', 'User6', 'User17', 'User10', 'User37']

User: User10
Top 20% Users (PCC): ['User30', 'User27', 'User32', 'User3', 'User41', 'User7', 'User44', 'User9', 'User42']
Top 20% Users (Discounted PCC): ['User30', 'User27', 'User32', 'User3', 'User41', 'User7', 'User44', 'User9', 'User42']

User: User39
Top 20% Users (PCC): ['User37', 'User46', 'User19', 'User13', 'User6', 'User41', 'User40', 'User24', 'User23']
Top 20% Users (Discounted PCC): ['User37', 'User46', 'User19', 'User13', 'User6', 'User41', 'User40', 'User24', 'User23']
```

```python
# 1.1.8: Compare predictions
for user in active_users:
    print(f"User: {user}")
    print("Predictions (No Bias):", predictions_no_bias[user])
    print("Predictions (With Discount):", predictions_with_discount[user])
    print()

#1.2.8: Compare the Results of Step 1.2.3 with Step 1.2.6
for user in active_users:
    print(f"User: {user}")
    print("Predictions (Cosine Similarity):", predictions_no_discount[user])
    print("Predictions (Discounted Similarity):", predictions_with_discount_centered[user])
    print()

#1.3.8: Compare Results of 1.3.3 and 1.3.6
for user in active_users:
    print(f"User: {user}")
    print("Predictions (PCC):", predictions_pcc[user])
    print("Predictions (Discounted PCC):", predictions_discounted_pcc[user])
    print()
```

```
User: User7
Predictions (No Bias): {'pull_up_bar': 2.4330905006848935, 'resistance_band': 2.2829753843352876}
Predictions (With Discount): {'pull_up_bar': 2.4330905006848935, 'resistance_band': 2.2829753843352876}

User: User10
Predictions (No Bias): {'exercise_bike': 2.6605422515346113, 'resistance_band': 3.4074361503605743, 'treadmill': 2.5872562762147564, 'yoga_mat': 2.5513433088072803}
Predictions (With Discount): {'exercise_bike': 2.6605422515346113, 'resistance_band': 3.4074361503605743, 'treadmill': 2.5872562762147564, 'yoga_mat': 2.5513433088072803}

User: User39
Predictions (No Bias): {'dumbbells': 2.30359371280898, 'elliptical': 2.4974677993325503, 'exercise_bike': 1.8323734461508043, 'foam_roller': 3.0798117935890255, 'kettlebell': 2.72320
Predictions (With Discount): {'dumbbells': 2.30359371280898, 'elliptical': 2.4974677993325503, 'exercise_bike': 1.8323734461508043, 'foam_roller': 3.0798117935890255, 'kettlebell': 2

User: User7
Predictions (Cosine Similarity): {'pull_up_bar': 1.970565948049961, 'resistance_band': 1.8449590750716671}
Predictions (Discounted Similarity): {'pull_up_bar': 1.970565948049961, 'resistance_band': 1.8449590750716671}

User: User10
Predictions (Cosine Similarity): {'exercise_bike': 2.663397540535851, 'resistance_band': 3.255055045468018, 'treadmill': 1.5428736914778372, 'yoga_mat': 2.2091744016719046}
Predictions (Discounted Similarity): {'exercise_bike': 2.663397540535851, 'resistance_band': 3.255055045468018, 'treadmill': 1.5428736914778372, 'yoga_mat': 2.2091744016719046}

User: User39
Predictions (Cosine Similarity): {'dumbbells': 2.2604622889220365, 'elliptical': 2.4655285898253427, 'exercise_bike': 1.8353992051967485, 'foam_roller': 3.0853543111930595, 'kettlebe
Predictions (Discounted Similarity): {'dumbbells': 2.2604622889220365, 'elliptical': 2.4655285898253427, 'exercise_bike': 1.8353992051967485, 'foam_roller': 3.0853543111930595, 'kett

User: User7
Predictions (PCC): {'pull_up_bar': 1.9705659480499615, 'resistance_band': 1.8449590750716671}
Predictions (Discounted PCC): {'pull_up_bar': 1.9705659480499615, 'resistance_band': 1.8449590750716671}

User: User10
Predictions (PCC): {'exercise_bike': 2.663397540535851, 'resistance_band': 3.2550550454680187, 'treadmill': 1.542873691477837, 'yoga_mat': 2.2091744016719046}
Predictions (Discounted PCC): {'exercise_bike': 2.663397540535851, 'resistance_band': 3.2550550454680187, 'treadmill': 1.542873691477837, 'yoga_mat': 2.2091744016719046}

User: User39
Predictions (PCC): {'dumbbells': 2.2604622889220365, 'elliptical': 2.4655285898253423, 'exercise_bike': 1.8353992051967485, 'foam_roller': 3.08535431119306, 'kettlebell': 2.626321585
Predictions (Discounted PCC): {'dumbbells': 2.2604622889220365, 'elliptical': 2.4655285898253423, 'exercise_bike': 1.8353992051967485, 'foam_roller': 3.08535431119306, 'kettlebell':
```

Part 2: Item-based Collaborative Filtering:

- **Discounted Similarity Impact**:
  - For **items like rowing_machine and pull_up_bar**, applying **discounted similarity** significantly narrowed down the **Top Recommended Items**, focusing only on the most relevant items.
  - Without **bias correction (No Discount)**, items like yoga_mat and dumbbells for rowing_machine, and rowing_machine and resistance_band for pull_up_bar, appeared more frequently in the top recommendations.
  - **With Discounting**, the results became more **refined**, emphasizing **relevant items** (e.g., yoga_mat for rowing_machine and rowing_machine for pull_up_bar). This suggests a **reduction in irrelevant or noisy suggestions**, enhancing recommendation quality.
- **Cosine Similarity and PCC with Bias Correction**:
  - Both **Cosine Similarity** and **PCC** applied in **Part 2** continue to focus on **comparative preferences** rather than raw rating scores. They both adjust for **user biases**, which helps in maintaining stable and consistent recommendations.

- The use of **similarity discounting** further refines the top recommendations, ensuring that only the most relevant and closely related items are prioritized.

Comparison and Comment on Results of Case Studies 2.1, 2.2, and 2.3:

```
#2.1.7: Compare Top Closest Items
for item in target_items:
    print(f"Item: {item}")
    print("Top 25% Items (No Bias):", top_25_items[item].index.tolist())
    print("Top 20% Items (With Discount):", top_20_items_discounted[item].index.tolist())
    print()

# 2.2.7: Compare Top 20% items using Cosine Similarity vs Discounted Similarity
for item in target_items:
    print(f"Item: {item}")
    print("Top 20% Items (No Bias):", top_20_items_centered[item].index.tolist())
    print("Top 20% Items (With Discount):", top_20_items_discounted_centered[item].index.tolist())
    print()

# 2.3.7: Compare Top 20% Closest Items using PCC vs Discounted PCC.
for item in target_items:
    print(f"Item: {item}")
    print("Top 20% Items (PCC):", top_20_items_pcc[item].index.tolist())
    print("Top 20% Items (Discounted PCC):", top_20_items_discounted_pcc[item].index.tolist())
    print()
```

```
Item: rowing_machine
Top 25% Items (No Bias): ['yoga_mat', 'dumbbells']
Top 20% Items (With Discount): ['yoga_mat']

Item: pull_up_bar
Top 25% Items (No Bias): ['rowing_machine', 'resistance_band']
Top 20% Items (With Discount): ['rowing_machine']

Item: rowing_machine
Top 20% Items (No Bias): ['pull_up_bar']
Top 20% Items (With Discount): ['pull_up_bar']

Item: pull_up_bar
Top 20% Items (No Bias): ['rowing_machine']
Top 20% Items (With Discount): ['rowing_machine']

Item: rowing_machine
Top 20% Items (PCC): ['pull_up_bar']
Top 20% Items (Discounted PCC): ['pull_up_bar']

Item: pull_up_bar
Top 20% Items (PCC): ['rowing_machine']
Top 20% Items (Discounted PCC): ['rowing_machine']
```

```python
#2.1.8: Compare Predictions
for item in target_items:
    print(f"Item: {item}")
    print("Predictions (No Bias):", predictions_items_no_bias[item])
    print("Predictions (With Discount):", predictions_items_with_discount[item])
    print()

#2.2.8: Comparison of Results from Point 2.2.3 and Point 2.2.6
for item in target_items:
    print(f"Item: {item}")
    print("Predictions (No Bias):", predictions_items_centered[item])
    print("Predictions (With Discount):", predictions_items_discounted_centered[item])
    print()

# 2.3.8: Compare Predictions using PCC vs Discounted PCC.
for item in target_items:
    print(f"Item: {item}")
    print("Predictions (PCC):", predictions_items_pcc[item])
    print("Predictions (Discounted PCC):", predictions_items_discounted_pcc[item])
    print()
```

```
Item: rowing_machine
Predictions (No Bias): {'User12': 2.546508169146003, 'User14': 3.968994553902664, 'User22': 1.0, 'User25': 1.0, 'User26': 1.0, 'User29': 3.968994553902664, 'User34': 4.0, 'User43': 2.968994553902664, 'User44': 5.0, 'User49': 3.0}
Predictions (With Discount): {'User12': 4.0, 'User14': 3.0, 'User22': 1.0, 'User25': 1.0, 'User26': 1.0, 'User29': 3.0, 'User34': 4.0, 'User43': 2.0, 'User44': 0, 'User49': 3.0}

Item: pull_up_bar
Predictions (No Bias): {'User1': 2.5123948415500106, 'User12': 3.0, 'User14': 5.0, 'User15': 2.0, 'User17': 3.0, 'User19': 4.5123948415500115, 'User21': 2.0, 'User25': 4.0, 'User27': 2.537184524650033, 'User29': 0, 'User31': 1.48760515
Predictions (With Discount): {'User1': 3.0, 'User12': 0, 'User14': 0, 'User15': 2.0, 'User17': 3.0, 'User19': 5.0, 'User21': 2.0, 'User25': 0, 'User27': 4.0, 'User29': 0, 'User31': 1.0, 'User33': 3.0, 'User34': 0, 'User35': 4.0, 'User3

Item: rowing_machine
Predictions (No Bias): {'User12': 0, 'User14': 0, 'User22': 1.0, 'User25': 0, 'User26': 3.0, 'User29': 0, 'User34': 0, 'User43': 0, 'User44': 5.0, 'User49': 2.0}
Predictions (With Discount): {'User12': 0, 'User14': 0, 'User22': 1.0, 'User25': 0, 'User26': 3.0, 'User29': 0, 'User34': 0, 'User43': 0, 'User44': 5.0, 'User49': 2.0}

Item: pull_up_bar
Predictions (No Bias): {'User1': 3.0, 'User12': 0, 'User14': 0, 'User15': 2.0, 'User17': 3.0, 'User19': 5.0, 'User21': 2.0, 'User25': 0, 'User27': 4.0, 'User29': 0, 'User31': 1.0, 'User33': 3.0, 'User34': 0, 'User35': 4.0, 'User39': 2.
Predictions (With Discount): {'User1': 3.0, 'User12': 0, 'User14': 0, 'User15': 2.0, 'User17': 3.0, 'User19': 5.0, 'User21': 2.0, 'User25': 0, 'User27': 4.0, 'User29': 0, 'User31': 1.0, 'User33': 3.0, 'User34': 0, 'User35': 4.0, 'User3

Item: rowing_machine
Predictions (PCC): {'User12': 0, 'User14': 0, 'User22': 1.0, 'User25': 0, 'User26': 3.0, 'User29': 0, 'User34': 0, 'User43': 0, 'User44': 5.0, 'User49': 2.0}
Predictions (Discounted PCC): {'User12': 0, 'User14': 0, 'User22': 1.0, 'User25': 0, 'User26': 3.0, 'User29': 0, 'User34': 0, 'User43': 0, 'User44': 5.0, 'User49': 2.0}

Item: pull_up_bar
Predictions (PCC): {'User1': 3.0, 'User12': 0, 'User14': 0, 'User15': 2.0, 'User17': 3.0, 'User19': 5.0, 'User21': 2.0, 'User25': 0, 'User27': 4.0, 'User29': 0, 'User31': 1.0, 'User33': 3.0, 'User34': 0, 'User35': 4.0, 'User39': 2.0, '
Predictions (Discounted PCC): {'User1': 3.0, 'User12': 0, 'User14': 0, 'User15': 2.0, 'User17': 3.0, 'User19': 5.0, 'User21': 2.0, 'User25': 0, 'User27': 4.0, 'User29': 0, 'User31': 1.0, 'User33': 3.0, 'User34': 0, 'User35': 4.0, 'User
```

Improvements:

1. **Hybrid Models**:
    a. While **user-based CF** and **item-based CF** are powerful individually, combining these two methods in a **hybrid recommender system** could leverage the strengths of both approaches. **Hybrid models** could better handle cold-start problems and provide more comprehensive recommendations by considering both user preferences and item relationships.
2. **Refining Discounting Techniques**:
    a. In **item-based CF**, the **discounting** mechanism showed a promising improvement in the quality of recommendations. However, more **advanced discounting techniques** could be explored. For instance,

incorporating **contextual or temporal factors** (such as adjusting similarity based on user activity or seasonal preferences) could enhance the personalization of recommendations further.

3. **Incorporating User and Item Attributes**:
    a. Including additional **metadata** (e.g., product features, user demographics) could improve the recommendation accuracy. By combining **content-based features** with **collaborative filtering**, a **hybrid recommendation approach** could provide even more accurate and personalized suggestions.
4. **Evaluation Metrics**:
    a. For future improvements, the use of more **robust evaluation metrics** such as **Mean Absolute Error (MAE)** or **Root Mean Square Error (RMSE)** would help to better assess the effectiveness of different weighting schemes and similarity measures in providing accurate predictions.

Conclusion:

In this assignment, we analyzed the impact of **significance weighting** and different **similarity measures** in both **user-based** and **item-based collaborative filtering (CF)** methods. The results from the case studies lead to several key conclusions.

In **Part 1**, which focused on **user-based CF**, we found that **Cosine Similarity** and **Pearson Correlation Coefficient (PCC)** produced very similar results, both with and without **bias adjustment (mean-centering)**. This indicates that adjusting for individual user biases is critical for accurate similarity calculation, but **discounting** did not significantly affect the **user similarity rankings** or **predicted ratings**. In contrast, **Part 2**, which dealt with **item-based CF**, showed that **discounting** had a more pronounced effect, significantly refining the recommendations. When **discounting** was applied, the **Top Recommended Items** became more focused on the most relevant items, highlighting how discounting helps improve recommendation quality by reducing the influence of less significant items.

The **bias adjustment techniques** (mean-centering) in both **Cosine Similarity** and **PCC** led to consistent similarity results across both **user-based** and **item-based** CF, reinforcing that adjusting for biases is crucial for accurate and stable recommendations. Additionally, **discounted similarity** helped improve the **item-based recommendations** by narrowing the list to the most relevant items.

In conclusion, **discounting** proved to be more effective in **item-based CF** by improving recommendation precision, while its impact was less significant in **user-based CF**. Future improvements could involve further refining discounting methods, exploring hybrid systems combining both user and item-based CF, and incorporating additional features or contextual data to enhance personalization and recommendation accuracy.

References:

- Comparison of User Based and Item Based Collaborative Filtering Recommendation Services
    - CollaborativeFiltering19JUNI.pdf
- Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison
    - Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison - ScienceDirect
- Novel Significance Weighting Schemes for Collaborative Filtering: Generating Improved Recommendations in Sparse Environments
    - (PDF) Novel Significance Weighting Schemes for Collaborative Filtering: Generating Improved Recommendations in Sparse Environments
- Cleaning Data from Web Sources: Techniques for Scraped Data
    - Cleaning Data from Web Sources: Techniques for Scraped Data
- Data Cleaning After Scraping: Why Do You Need It
    - Data Cleanup After Scraping: Steps and Tools to Use