# AIE425: Intelligent Recommender systems, Fall Semester 24/25

## Assignment #1: Neighborhood CF models (user, Item-based CF)

### Name: Mohamed Ahmed Fathi

### ID: 221101699

**Customer Feedback and Rating Type on Amazon**

**Introduction**

In designing a recommender system for Amazon, understanding how Amazon collects and interprets customer feedback is essential. Unlike platforms that rely solely on implicit data, Amazon combines explicit numeric ratings with user interactions and behaviors. This mix of feedback types enables Amazon to develop effective recommendation systems for products based on user preferences and activity.

**Customer Feedback and Rating Types**

Amazon collects both explicit and implicit feedback from its users, providing detailed insights into customer preferences and behaviors. The primary types of feedback on Amazon are as follows:

1. **Star Ratings and Reviews**:
   * **Star Ratings**: Amazon allows customers to rate products on a scale of 1 to 5 stars. These ratings are explicit feedback, with higher star ratings (4-5) typically indicating a strong preference for a product.
   * **Text Reviews**: Customers can leave detailed reviews about their experience. While not always quantified, reviews offer valuable context about why a product received a certain rating, adding a qualitative aspect to feedback.
2. **Purchase Behavior**:
   * **Purchases and Repurchases**: Purchasing a product is a strong signal of interest. Repeat purchases further reinforce positive feedback, indicating satisfaction and preference for a specific item or brand.
   * **Add to Cart/Wishlist**: Adding an item to the cart or wishlist signals intent to purchase, often suggesting positive interest even if the purchase doesn’t immediately follow.
3. **Browsing and Search Activity**:
   * **Product Views**: Amazon records how often a user views a product page. Repeated views or long time spent on certain pages can imply interest, although this remains implicit feedback.
   * **Search Queries**: Specific search terms and frequency indicate customer interest and preferences, with more focused search queries often signaling higher intent.
4. **Abandonment and Returns**:
   * **Cart Abandonment**: Adding a product to the cart but not completing the purchase might indicate indecision or a lower preference level.
   * **Returns**: Returning a purchased product often indicates dissatisfaction, serving as an implicit form of negative feedback.
5. **Product Interactions**:
   * **“Helpful” Votes on Reviews**: Amazon allows users to mark reviews as helpful or unhelpful, indicating agreement with positive or negative feedback from other customers. This feedback contributes to the product’s overall perception in the community.

**Mapping Feedback to a Rating Scale**

To develop a recommender system for Amazon, both explicit and implicit feedback can be represented on a rating scale. Below is a possible mapping of feedback types to a 1–5 rating scale that quantifies user preferences.

| **Rating** | **Interaction Type** | **Description** |
| --- | --- | --- |
| 5 | Highly Positive | 5-star ratings, repeat purchases, “add to wishlist” action. |
| 4 | Positive | 4-star ratings, frequent views, multiple items from the same brand in the cart. |
| 3 | Neutral | 3-star ratings, occasional views, “add to cart” without purchase. |
| 2 | Low | 2-star ratings, brief views without adding to cart or purchase, occasional returns. |
| 1 | Negative | 1-star ratings, high return rate, or abandonment after brief viewing. |

**Data Preprocessing and Cleaning**

For this project, we utilize the Amazon dataset, which includes user\_id, name (product name), main\_category, sub\_category, image, link, ratings, discount\_price, and actual\_price. The following preprocessing steps ensure data quality and usability for a recommender system:

1. **Data Loading and Initial Inspection**  
   We loaded the Amazon dataset and reviewed its structure, focusing on essential columns for recommendation, such as user\_id, name, and ratings.
2. **Handling Missing Values**  
   Missing values in the user\_id, name, or ratings columns were removed, as these fields are critical for building a user-item matrix. Non-essential fields with missing values were either retained as-is or removed based on relevance.
3. **Removing Duplicates**  
   Duplicate entries (e.g., if a user rated the same product multiple times) could skew recommendation results, so duplicates based on user\_id and name were identified and removed.
4. **Standardizing Ratings**  
   To ensure consistency in feedback interpretation, we confirmed that ratings in the ratings column were in a 1–5 scale, as this is required for collaborative filtering algorithms.
5. **Handling Price Information**  
   discount\_price and actual\_price were retained for potential future analysis but were not directly relevant for the recommendation system.

**Creating the User-Item Matrix**

The user-item matrix is a core structure used for collaborative filtering, representing each user’s rating of various products.

1. **Matrix Structure**
   * **Rows**: Each row represents a unique user\_id.
   * **Columns**: Each column corresponds to a unique name, representing a product.
   * **Values**: The values in the matrix are user ratings, which reflect how a user rated each product.
2. **Matrix Construction**  
   Using a pivot table, we transformed the dataset into the user-item matrix format, setting user\_id as rows, name as columns, and ratings as values.
3. **Handling Missing Ratings**  
   Due to the sparsity of user-product interactions, many cells in the matrix are empty (no rating exists for a particular user-product pair). Missing values (NaN) can be handled in several ways:
   * **Sparse Matrix**: We can leave the matrix sparse for algorithms that handle missing values, such as matrix factorization techniques.
   * **Imputation**: Alternatively, we may fill missing ratings with the mean product rating or a default value of 0, depending on the algorithm used.

**Understanding Cosine Similarity and Pearson Correlation**

* **Cosine Similarity**: Measures the cosine of the angle between two vectors, disregarding magnitude and focusing on orientation. In a user-item matrix, cosine similarity helps in understanding the similarity between users’ or items’ ratings based solely on the direction of preferences, independent of rating magnitude.
* **Pearson Correlation**: Calculates the correlation coefficient between two sets of data, capturing both direction and magnitude. Pearson correlation normalizes ratings, making it useful when users have different rating scales.

**Example Calculation**

Given the matrix snippet:

| **User** | **Samsung AC (1.5 Ton)** | **Voltas AC (1.5 Ton)** | **Samsung AC (1 Ton)** | **LG AC (1 Ton)** |
| --- | --- | --- | --- | --- |
| Babs\_05 | 4.1 | NaN | 4.0 | 4.2 |
| Orlenay | 5.0 | 2.6 | 3.0 | 4.3 |
| massdosage | 4.0 | 3.5 | 2.4 | 3.4 |
| Knapster01 | 5.0 | 2.5 | 3.4 | 2.1 |
| eartle | 3.0 | 2.0 | 4.0 | 2.6 |

**Step-by-Step Comparison**

1. **Cosine Similarity Calculation**:
   * **Interpretation**: Cosine similarity between users measures how similar the direction of their ratings is, without considering individual user biases (e.g., whether one user rates generally high or low).
   * **Use Case Strength**: Useful when the scale of ratings differs widely among users but the relative pattern of preferences remains consistent.
   * **Example**: If Babs\_05 and Orlenay both rate the Samsung 1.5 Ton and LG AC products highly but differ in rating magnitude, cosine similarity will still identify them as similar.
2. **Pearson Correlation Calculation**:
   * **Interpretation**: Pearson correlation focuses on the relationship between ratings after normalizing by each user’s average rating. This makes it better suited for scenarios where users differ in rating habits (e.g., one user rates everything highly, another conservatively).
   * **Use Case Strength**: Appropriate when users’ overall rating scales vary widely, as it removes individual biases.
   * **Example**: In the matrix, if massdosage consistently rates products lower than Orlenay but shares a similar relative preference pattern, Pearson correlation will reflect their similarity better than cosine similarity.
3. **Pros and Cons of Each Method**

| **Method** | **Pros** | **Cons** |
| --- | --- | --- |
| **Cosine Similarity** | Ignores individual biases, focusing on relative preference direction. | May overstate similarity for users with different rating scales but similar preference order. |
| **Pearson Correlation** | Accounts for rating scale differences, reducing individual rating biases. | Requires more data (multiple ratings per user) to accurately calculate averages. |

1. **Conclusion**

* **Cosine Similarity**: Best for datasets where we are interested in identifying users with similar patterns, regardless of absolute rating scales.
* **Pearson Correlation**: More effective for datasets with substantial variation in users' rating habits, as it normalizes ratings.

For the Amazon recommender system, **Pearson correlation** may provide better insights, as users often have varied rating scales.