In our project, two recommendation algorithms used in the e-commerce project. These algorithms aim to enhance user experience by delivering personalized product recommendations. The first algorithm employs Collaborative Filtering using Singular Value Decomposition (SVD), while the second utilizes Item-Based Collaborative Filtering with Cosine Similarity. Both methods leverage user-product interaction data to generate relevant recommendations effectively.

**Collaborative Filtering Using SVD**

Collaborative filtering predicts a user's preference for a product based on the preferences of other users. Singular Value Decomposition (SVD) is used to decompose the user-product interaction matrix into latent features, enabling the prediction of missing interactions (e.g., ratings).

**Implementation Steps**

1. **Data Preprocessing:**
   * The dataset contains user actions such as click, view, and buy.
   * Actions are mapped to numerical values:
     + Click = 1
     + View = 0.5
     + Buy = 2.
2. **Dataset Preparation:**
   * The preprocessed data is converted into the Surprise library’s dataset format using the Reader class.
   * The Dataset.load\_from\_df() method is employed to create a dataset suitable for training.
3. **Training the Model:**
   * The data is split into training and testing sets using train\_test\_split().
   * An SVD model is trained on the training set to learn latent features for users and products.
4. **Generating Recommendations:**
   * For a specific user, the model predicts ratings for all products.
   * Products are sorted by predicted ratings, and the top-N products are recommended.

**Item-Based Collaborative Filtering Using Cosine Similarity**

Item-based collaborative filtering calculates similarity between items based on user interactions. It then recommends products similar to those the user has interacted with. Cosine similarity is used to measure the degree of similarity between item vectors.

**Implementation Steps**

1. **Data Preprocessing:**
   * Actions (click, view, buy) are encoded into numerical values:
     + Click = 1
     + View = 1
     + Buy = 2.
2. **Creating an Interaction Matrix:**
   * A pivot table is created with users as rows, products as columns, and interaction values as matrix elements.
   * Missing values are filled with zero to represent no interaction.
3. **Calculating Similarities:**
   * Cosine similarity is computed between the columns (products) of the interaction matrix to create a product similarity matrix.
4. **Generating Recommendations:**
   * For a specific user, the algorithm identifies products they have interacted with.
   * Scores are calculated for other products based on their similarity to interacted products and the user's interaction strength.
   * Products are ranked by scores, and the top-N products are recommended.

**4.4 Comparison of Algorithms**

| **Feature** | **SVD Collaborative Filtering** | **Item-Based Collaborative Filtering** |
| --- | --- | --- |
| **Similarity Source** | Based on user-product interaction patterns. | Based on co-occurrence in user interactions. |
| **Data Requirements** | Requires user-product interaction matrix. | Requires user-product interaction matrix. |
| **Cold-Start Problem** | Affected for new users/products. | Affected for new users/products. |
| **Scalability** | Computationally expensive for large datasets. | Computationally intensive for large datasets. |
| **Personalization** | Highly personalized. | Personalized based on similar items. |

The recommendation system in this e-commerce project integrates two algorithms:

1. **Collaborative Filtering with SVD:** This method provides highly personalized recommendations by leveraging latent features of users and products.
2. **Item-Based Collaborative Filtering with Cosine Similarity:** This method identifies similar products based on user interaction patterns.

While both approaches have strengths and limitations, their integration offers a robust recommendation engine. Future work could involve developing a hybrid model combining these techniques to maximize recommendation accuracy and address cold-start problems effectively.

**Presentation Slides: Implementation of Recommendation Algorithms**

**Slide 1: Algorithm Description - User-Based Collaborative Filtering with SVD**

**Title:** User-Based Collaborative Filtering with SVD

**Key Points:**

* **Overview:**
  + Combines User-Based Collaborative Filtering with Singular Value Decomposition (SVD).
  + Recommends products based on user interaction history and predicted preferences.
* **Data Preparation:**
  + Input Dataset:
    - User-product interactions recorded as actions: click, view, buy.
    - Actions mapped to numerical values: click = 1, view = 0.5, buy = 2.
  + Format Conversion:
    - Data converted for Surprise library using a 0 to 2 rating scale.
* **Model Training:**
  + Split data into training and testing sets.
  + Train SVD to decompose user-item matrix into latent factors.
  + Identify patterns in user preferences and product characteristics.
* **Generating Recommendations:**
  + Predict ratings for all unseen products for a user.
  + Sort predictions to prioritize products with the highest scores.
  + Select the top 10 products as recommendations.
* **Deployment:**
  + Save the trained SVD model using Joblib for reuse.
  + Load the model at runtime for real-time recommendations.

**Slide 2: Algorithm Description - Item-Based Collaborative Filtering**

**Title:** Item-Based Collaborative Filtering with Cosine Similarity

**Key Points:**

* **Overview:**
  + Recommends products similar to those the user interacted with.
  + Uses cosine similarity to measure relationships between products based on user interactions.
* **Data Preparation:**
  + Input Dataset:
    - User-product interactions recorded as actions: click, view, buy.
    - Actions mapped to numerical values: click = 1, view = 1, buy = 2.
  + Interaction Matrix:
    - Create a matrix with users as rows, products as columns, and interactions as values.
    - Fill missing values with zeros.
* **Model Training:**
  + Calculate cosine similarity between product vectors.
  + Generate a product similarity matrix to capture relationships.
* **Generating Recommendations:**
  + Identify products the user interacted with.
  + Score other products based on similarity and interaction strength.
  + Recommend the top-N ranked products.
* **Deployment:**
  + The similarity matrix is precomputed and used during runtime to quickly generate recommendations.