**TASK-4**

RECOMMENDATION SYSTEM

Create a simple recommendation system that suggests items to users based on their preferences. You can use techniques like collaborative filtering or content-based filtering to recommend movies, books, or products to users

**Introduction**

* Recommender systems leverage data and programming to suggest relevant products, services, or content to users.
* Examples include Netflix, Amazon, and Uber Eats, all personalizing user experiences.
* Python is a powerful tool for building these systems due to its extensive libraries and machine learning frameworks.

**Types of Recommendation Systems**

* **Content-Based Filtering:** Recommends items similar to those a user has liked previously. It analyzes item properties and user profiles to find matches.
  + **Item Profile:** A representation of an item's key characteristics. For movies, this might include genre, director, actors, etc.
  + **User Profile:** A summary of a user's preferences based on their past interactions.
  + **TF-IDF:** A method to evaluate a word's significance within a document or corpus. It considers both a word's frequency and its rarity.
* **Collaborative Filtering:** Recommends items based on the preferences of similar users. It assumes users with similar tastes will like similar things.
  + **User-Item Matrix:** A table representing user ratings or interactions with items.
  + **Similarity Measures:** Techniques to quantify how similar users or items are. Common methods include cosine similarity and Pearson correlation.

**Implementation Steps**

1. **Importing Libraries:** Python libraries like NumPy, Pandas, scikit-learn, Matplotlib, and Seaborn are imported for data manipulation, visualization, and machine learning tasks.
2. **Loading Datasets:** Two datasets are loaded:
   * User ratings for movies
   * Movie metadata like titles and genres
3. **Statistical Analysis of Ratings:** This code calculates various statistics about the ratings data, including:
   * Number of ratings, unique movies, and unique users
   * Average ratings per user and per movie
   * User rating frequency (number of ratings per user)
4. **Movie Rating Analysis:** This section analyzes movie ratings to find:
   * Lowest and highest rated movies
   * Number of users who rated these movies (provides insight into popularity)
5. **User-Item Matrix Creation:** A user-item matrix is a core structure in recommendation systems. It represents user ratings for items in a sparse matrix format.
6. **Movie Similarity Analysis:** This code uses the K-Nearest Neighbors (KNN) algorithm to find movies similar to a given movie. It considers factors like genre, actors, or directors.
7. **Movie Recommendation with respect to User Preference:** This function recommends movies to a specific user based on their highest-rated movie. It finds similar movies using KNN and suggests them to the user.

**Conclusion**

* Python recommendation systems personalize user experiences by suggesting relevant content.
* These systems leverage collaborative filtering, content-based filtering, or a combination of both (hybrid techniques).
* Machine learning and data-driven insights allow these systems to improve and adapt over time.
* Recommendation systems are crucial for various industries, boosting user satisfaction, engagement, and business growth.