Book Recommendation involves suggesting books to users depending on their past interest. Here I have used Item based collaborative filtering to identify books similar to a particular book.

I downloaded the dataset from <http://www2.informatik.uni-freiburg.de/~cziegler/BX/>

There are 3 csv files in the dataset:

1. BX-Books: This gives the book titles, ISBN and some other details about book.
2. BX-Users: This gives the user-id, location and their age who rated the books.
3. BX-Book-Ratings: This gives the ISBN of books, ratings for each book and user id who rated that particular book.

I have used Python and the Jupyter notebook for development.

I have used the following methods:

1. **K Nearest Neighbors (KNN) algorithm**

I used KNN because the data is categorical and as we know for categorical data we use classification which belongs to supervised learning. KNN algorithm is a family of popular non-probabilistic CF techniques. This technique computes the similarity between either users (user-to-user based CFs) or items (item-to-item based CFs) that are represented by columns or rows in a user-item matrix with similarity metrics such as cosine, adjusted cosine, or euclidean. This algorithm then predicts unknown ratings based on known ratings of similar users or items. The similarity values "provide the means to give more or less importance to these neighbors in the prediction”. In BX datasets kNN is a machine learning algorithm to find clusters of similar users based on common book ratings, and make predictions using the average rating of top-k nearest neighbors. The algorithm I used to compute the nearest neighbors is “brute”, and I specify “metric=cosine” so that the algorithm will calculate the cosine similarity between rating vectors.

1. **SVD algorithm**

SVD belongs to unsupervised learning. I chose SVD because I read that SVD works perfect in recommendation systems. Simple versions of recommendation systems compute similarity between items or people. More advanced methods use the SVD to create a theme space from the data and then compute similarities in the theme space.

Also, in the dataset there are missing values. As a recommendation system, I would like to know what the values of the missing entries would be if the user were to actually rate a particular book. However, because users don’t tend to read many books, the user-book matrix is approximately 98% sparse. To alleviate this issue of sparsity, UV-decomposition is an algorithm that uses dimensionality reduction, which finds a way to represent the matrix using a much smaller number of values than the total number of entries. Broadly dimensionality reduction would “remove unrepresentative or insignificant users or items to reduce the dimensions of the user-item matrix directly”. UVD is an instance of a more general theory called SVD (singular-value decomposition).

Matrix Factorization is simply a mathematical tool for playing around with matrices. The Matrix Factorization techniques are usually more effective, because they allow users to discover the latent (hidden)features underlying the interactions between users and items (books). I used singular value decomposition (SVD) — one of the Matrix Factorization models for identifying latent factors.

I compared two different algorithms in order to see if there are similarities in the results.

<https://github.com/Calliope-Maria/book-recommendation/blob/main/BookRecommendation.ipynb>