Book Recommendation System

„A recommendation engine is a class of machine learning which offers relevant suggestions to the customer.“

Types Of Recommendation Systems

1) **Content-based Filtering** - The algorithm recommends a product that is similar to those which a user has already read. In simple words, in this algorithm, we try to find a similar item.
2) **Collaborative-based Filtering** - Collaborative-based filtering recommendation systems are based on past interactions of users and target items. In simple words here, we try to search for similar customers and offer products based on what his or her lookalike has chosen.
3) **Hybrid Filtering Method** – This is basically a combination of both of the above methods. It is a complex model which recommends products based on the user’s history as well as similar users preferences.

Datasets

- **Books** – Dataset which contains all the information related to books like an author, title, year of publication, etc.
- **Users** – The second dataset contains registered users' information like user id or location.
- **Ratings** – Ratings dataset contains information like what rating a particular user has given to a certain book.

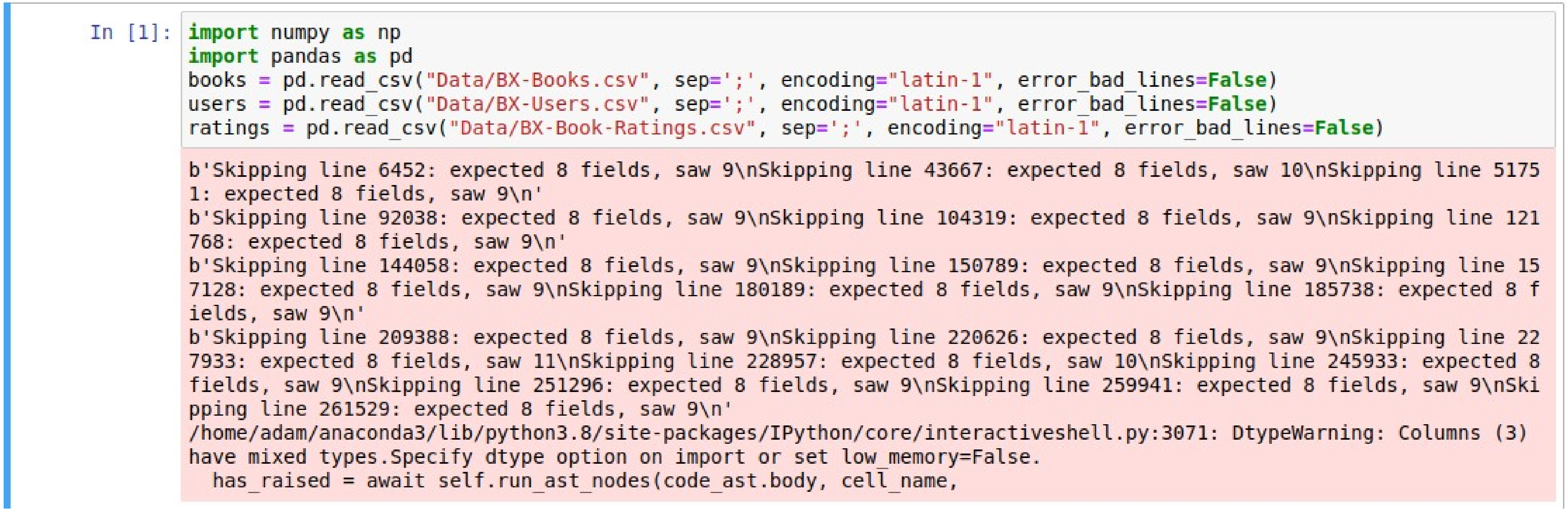
Loading The Datasets

While loading the files, we encounter following **problems**:

- The values in the CSV file are separated by semicolons, not by commas.
- There are some lines which do not work. We cannot import them using pandas and they throw us an error.
- Encoding of a file is in Latin

So while loading the data we have to handle for these exceptions and after running the code on the next slide we will get some warnings. These will show us the lines that have the error and we will know that these lines have been skipped while loading.

Loading The Datasets

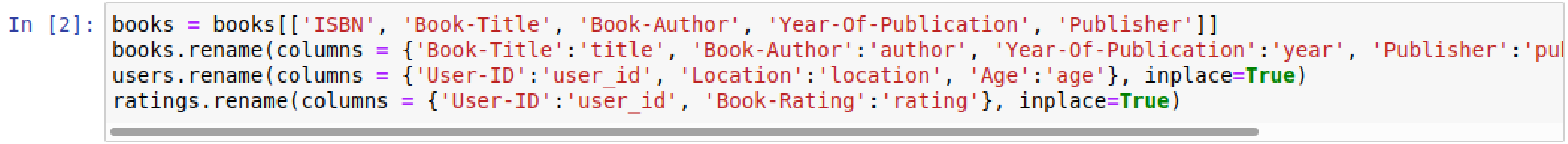


```
In [1]: import numpy as np
        import pandas as pd
        books = pd.read_csv("Data/BX-Books.csv", sep=';', encoding="latin-1", error_bad_lines=False)
        users = pd.read_csv("Data/BX-Users.csv", sep=';', encoding="latin-1", error_bad_lines=False)
        ratings = pd.read_csv("Data/BX-Book-Ratings.csv", sep=';', encoding="latin-1", error_bad_lines=False)
```

```
b'Skipping line 6452: expected 8 fields, saw 9\nSkipping line 43667: expected 8 fields, saw 10\nSkipping line 5175
1: expected 8 fields, saw 9\n'
b'Skipping line 92038: expected 8 fields, saw 9\nSkipping line 104319: expected 8 fields, saw 9\nSkipping line 121
768: expected 8 fields, saw 9\n'
b'Skipping line 144058: expected 8 fields, saw 9\nSkipping line 150789: expected 8 fields, saw 9\nSkipping line 15
7128: expected 8 fields, saw 9\nSkipping line 180189: expected 8 fields, saw 9\nSkipping line 185738: expected 8 f
ields, saw 9\n'
b'Skipping line 209388: expected 8 fields, saw 9\nSkipping line 220626: expected 8 fields, saw 9\nSkipping line 22
7933: expected 8 fields, saw 11\nSkipping line 228957: expected 8 fields, saw 10\nSkipping line 245933: expected 8
fields, saw 9\nSkipping line 251296: expected 8 fields, saw 9\nSkipping line 259941: expected 8 fields, saw 9\nSki
pping line 261529: expected 8 fields, saw 9\n'
/home/adam/anaconda3/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3071: DtypeWarning: Columns (3)
have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Preprocessing Data

- In the books dataset, we have some extra columns which are not required for our task (such as image URLs) so we are going to remove them.
- We will rename the columns in each dataset to make it easier to use them.



```
In [2]: books = books[['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher']]
        books.rename(columns = {'Book-Title':'title', 'Book-Author':'author', 'Year-Of-Publication':'year', 'Publisher':'pul
        users.rename(columns = {'User-ID':'user_id', 'Location':'location', 'Age':'age'}, inplace=True)
        ratings.rename(columns = {'User-ID':'user_id', 'Book-Rating':'rating'}, inplace=True)
```

We have **271,360** books data and total number of registered users on the website is approximately **278,000**. These users have given **over one million** ratings. Therefore, we can say that the dataset we have is very nice and reliable.

Defining The Problem

Our goal is to find user A who has liked and read book x and y, and user B who has also liked and read book x and y. Now if user A reads a new book z and likes it, we can recommend this book to user B. This is called **collaborative filtering**.

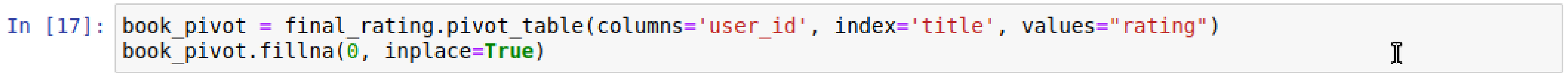
We can achieve this if we use **Matrix Factorization.** We will create a matrix where **columns** will be **users**, **indexes** will be **books** and **values** will be **individual ratings**.

For this kind of approach to work we need to **limit** which users and which books we are going to use. We will only consider users who rated more than **200** books and books that have been rated at least **50** times. This way the system will work as intended.

After applying the stated limitations, we created our final dataset that contains only the users who rated more than 200 books and books that have been rated at least 50 times. The shape of the final dataset is **59,850** rows and **8** columns.

Creating The Matrix

We will create a pivot table where columns will be user ids, the index will be book title and the value is ratings. If a user has not rated a particular book, we will substitute his rating value with the value NAN.

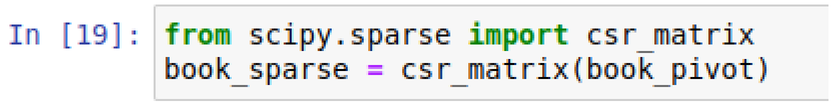


```
In [17]: book_pivot = final_rating.pivot_table(columns='user_id', index='title', values="rating")
         book_pivot.fillna(0, inplace=True)
```

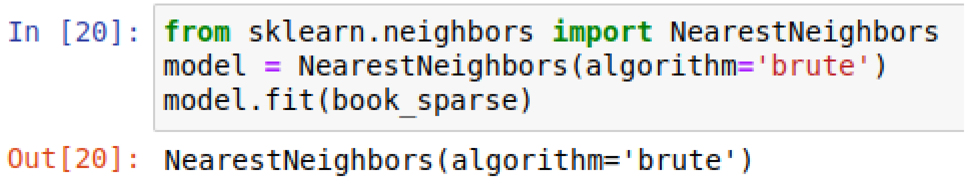
Modelling

We have prepared our dataset for modelling. We will use the nearest neighbors algorithm which is the same as K nearest that is used for clustering based on euclidian distance.

Our pivot table has a lot of zero values which is going to increase computing power needed for the algorithm We are going to convert the pivot table to a sparse matrix and then feed it to the model.



```
In [19]: from scipy.sparse import csr_matrix
         book_sparse = csr_matrix(book_pivot)
```

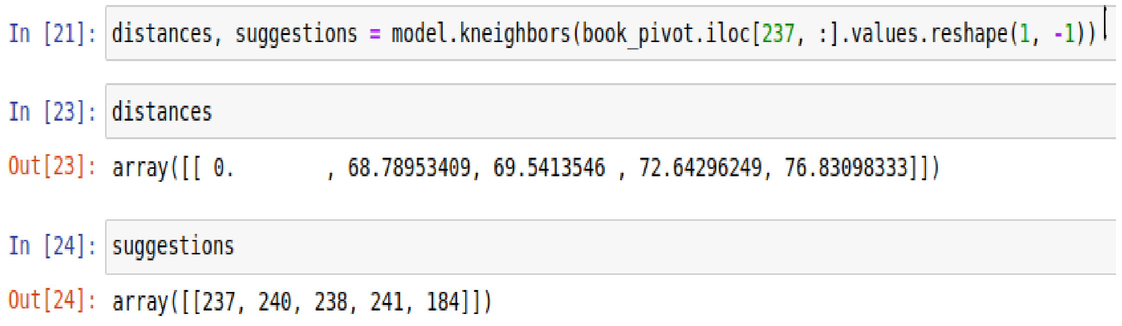


```
In [20]: from sklearn.neighbors import NearestNeighbors
         model = NearestNeighbors(algorithm='brute')
         model.fit(book_sparse)

Out[20]: NearestNeighbors(algorithm='brute')
```

Making a Prediction

Let’s make a prediction and see whether it is suggesting books or not. We will find the nearest neighbors to the input book id and after that, we will print the nearest 5 books. We are going to pass Harry Potter which is at index 237. This will provide us with distance and book id at that distance.



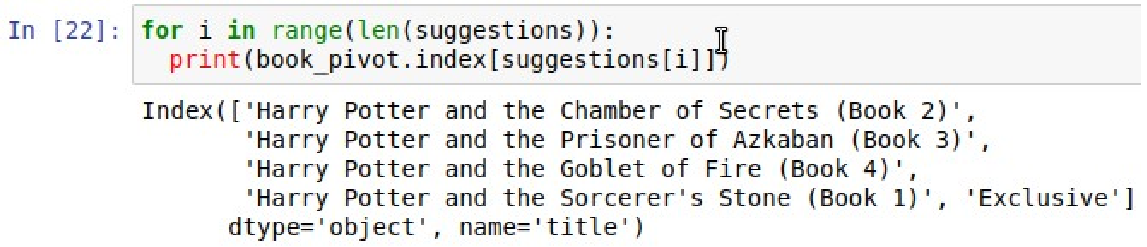
```
In [21]: distances, suggestions = model.kneighbors(book_pivot.iloc[237, :].values.reshape(1, -1))

In [23]: distances

Out[23]: array([[ 0.        , 68.78953409, 69.5413546 , 72.64296249, 76.83098333]])

In [24]: suggestions

Out[24]: array([[237, 240, 238, 241, 184]])
```



```
In [22]: for i in range(len(suggestions)):
           print(book_pivot.index[suggestions[i]])

         Index(['Harry Potter and the Chamber of Secrets (Book 2)',
                'Harry Potter and the Prisoner of Azkaban (Book 3)',
                'Harry Potter and the Goblet of Fire (Book 4)',
                'Harry Potter and the Sorcerer's Stone (Book 1)', 'Exclusive']
               dtype='object', name='title')
```