Task 3

This task focuses on building a **Recommendation System** to predict book ratings and suggest similar books based on descriptions. This system will predict ratings for books not yet read by a user and provide a ranked list of recommended books they might be interested in. The steps are as follows:

1. **Data Selection**: Use the BookId, Description columns, filtering only rows where the language is "English".
2. **TF-IDF Representation**: Create a **TF-IDF (Term Frequency-Inverse Document Frequency)** matrix for unigrams (single words) and bigrams (pairs of words) in the description column, using the stop\_words parameter of the **TfidfVectorizer** to remove common stopwords.
3. **Similarity Calculation**: For each book, compute the cosine similarity with every other book and store the top 100 most similar books in a Python dictionary.
4. **Recommendation Function**: Implement a function recommend(item\_id, num) that takes a book ID and a number N as input, and returns the top N most similar books to the given book ID.

Task 4

The task involves classifying book genres based on descriptions. First, we need to filter books belonging to the 10 most frequent genres and clean the description column. Then, apply **Word2Vec** to generate 200-300 dimensional embeddings for each description, saving them as .pkl files using **pickle**. Split the data into training (80%) and testing (20%) sets using train\_test\_split().

Next, train three classification models on the training data: **Naive Bayes**, **Support Vector Machines (SVM)**, and **Random Forests**. Evaluate the models using **Precision, Recall, F-Measure**, and **Accuracy** metrics. Perform **10-fold Cross Validation** to assess model performance and create a table summarizing the results for each method and parameter.

Due to limited resources and time for training, while the process and some training steps are displayed, in the evaluations with the classifiers we chose to use a pretrained model.

Initially we clean and prepare the data

We divide the dataset into 2 sets: train & test. If it has already been split earlier, data from data can be loaded saved files.

The next step is to train the training set using the word2vec model. If they has already been trained earlier, the model can be loaded from saved file. For training we will implement an iterative approach, where we will check the model at each epoch to monitor progress and avoid overfitting. At the end, the optimal model is automatically selected.

BONUS

This task focuses on **Content-Based Image Retrieval (CBIR)**, which uses visual features (color, texture, shape) to search for similar images. The steps are as follows:

1. **Download Images**: Obtain around 400-500 book cover images from the "Best Books Ever" dataset.
2. **Calculate Histograms**: Use OpenCV’s cv2.calcHist() to compute color histograms for each image, separating them by RGB channels.
3. **Create DataFrame**: Combine the histograms into a DataFrame with image IDs for easy reference.
4. **Test Image**: Select one test image and calculate its histogram.
5. **Similarity Search**: Compare the histograms of the test image with the dataset using **Euclidean Distance** or **Cosine Similarity** to find the 4 most similar images.
6. **Visualization**: Use Matplotlib to display the search results.

For this query 500 random images have been selected from the database and are located in the img folder

Calculations cosine similarity using dictionary

The images for testing are placed in the img\_test folder