Indie Book Recommender

Nour Rachdi, Jan Thurner, Selin Demirtürk

October 27, 2025

Contents

1 Introduction

Our project involves analyzing books and authors to examine their popularity, the factors that contribute to their popularity, and identifying indie authors to discover hidden and underrated gems. We built an Indie book recommender that allows users to select one or more books they enjoyed reading in the past, and the system recommends books that are similar to their preferences while highlighting titles from small authors.

We analyzed the data with respect to three key principles. The first goal was to determine whether there are ways to identify indie authors based on specific metrics within the data. Second, we compared different books to find similar books to one or more of the users' preferred books. We also investigated whether there are authors, genres, and books that are more popular than others, and what might be the reasons for this.

2 Data collection and cleaning

We used two Kaggle datasets as our primary data sources [1]: an Amazon books dataset containing book metadata, and a corresponding reviews dataset with user ratings and text reviews. Together, they contained the core information needed for our recommender system: titles, authors, publishers, genres, descriptions, cover URLs, and individual user reviews with ratings. The books dataset included over 212,000 entries, while the reviews dataset contained approximately 3 million individual reviews.

For the data cleaning part, we focused on ensuring data quality for the key variables. We dropped all entries with missing values of our main variables (author names, book titles and book descriptions) and removed some unnecessary or redundant columns that did not contribute to our analysis. We also added several features to support our recommendation system. We computed aggregate metrics including: average ratings, number of reviews per book, and the most critical addition: the "is_indie" binary flag to classify authors as either independent or mainstream. This classification combined information from both datasets. An indie author meets the following requirements: self-published and has less than 3 published books or fewer than 20 total reviews across their works. The final tables were the books table, the reviews table, and the authors table, with the latter combining information from the first two.

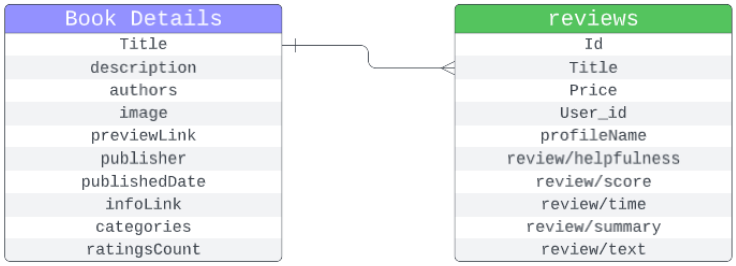


Figure 1: Amazon books and reviews dataset structure

Given that the datasets are very large in size, they are stored in a Google Drive folder, and are downloaded automatically once when the application initializes.

3 EDA and visualizations

During the Exploratory Data Analysis, we investigated multiple aspects. First, we analyzed the data for completeness, then we investigated it to identify well-known authors, and lastly, we analyzed different genres in terms of their popularity.

First, the analysis showed that after cleaning, the Data, consisting of three tables (Books, Reviews, Authors), is missing only few values. The missing values are mostly images of the books, which are not required for the data analysis, but for a nice UI of the recommender. Figure 2 highlights these findings. The review-table and the author-table are missing even less entries, therefore these plots are not included in this report.

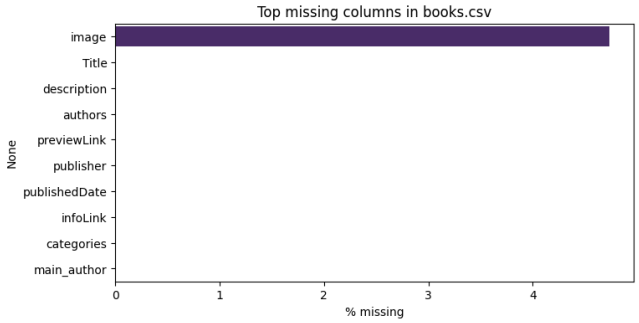
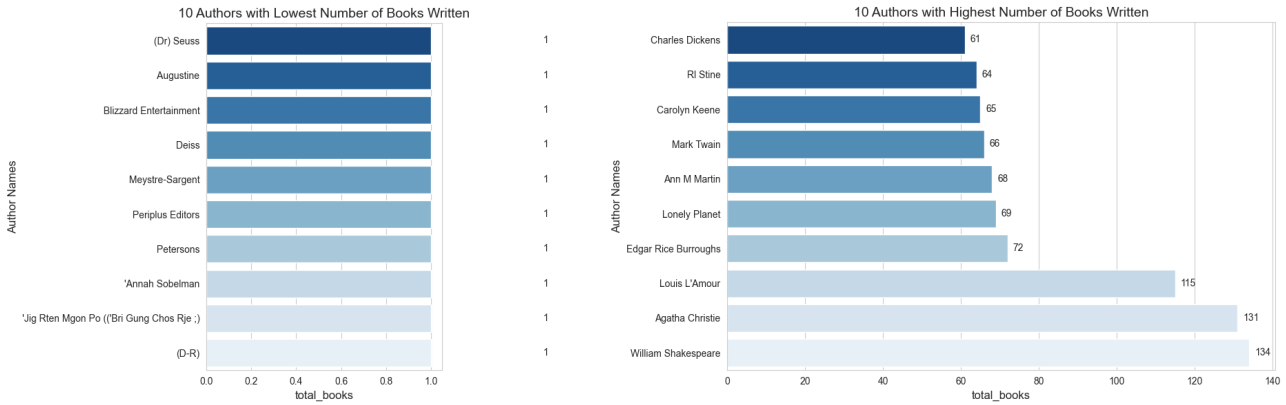


Figure 2: Columns with most missing values after data cleaning

When analyzing the authors, we find that authors who already have many publications are more widely known than authors who have not yet published many books. There are a lot of unknown authors who have only published one book. The more books authors have published, the more well-known they become and the thinner the field of authors with a similar number of publications gets. Figure 3 shows this difference.

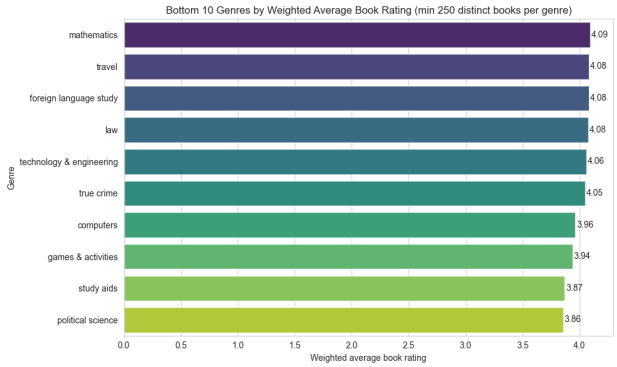


(a) Lowest number of books written

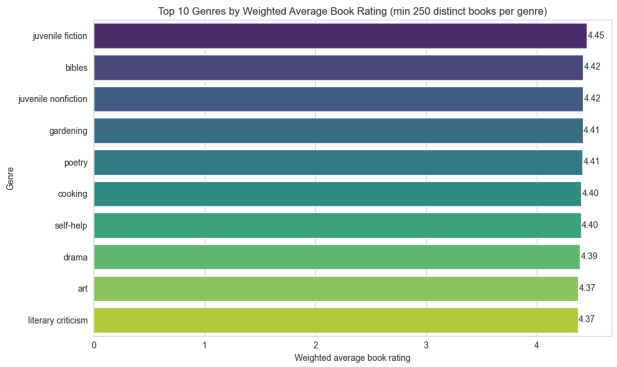
(b) Highest number of books written

Figure 3: Comparison of authors who published only a few books to authors who published a lot.

The analysis of more than 50 genres with more than 250 books published each yields the result that the different genres are similar in popularity. Only the bottom four genres see a decrease in reader ratings.



(a) Lowest rated genres out of 54 analyzed genres



(b) Highest rated genres out of 54 analyzed genres

Figure 4: Comparison of lowest versus highest rated genres with more than 250 books per genre.

While the user ratings for different genres are fairly equally distributed, we do see a clear trend in the number of published books per genre. With over 20.000 books, the fantasy market is so far the biggest one. History and Religion are the second and third biggest ones and have approximately one third of the books of fantasy each, as illustrated in figure 5.

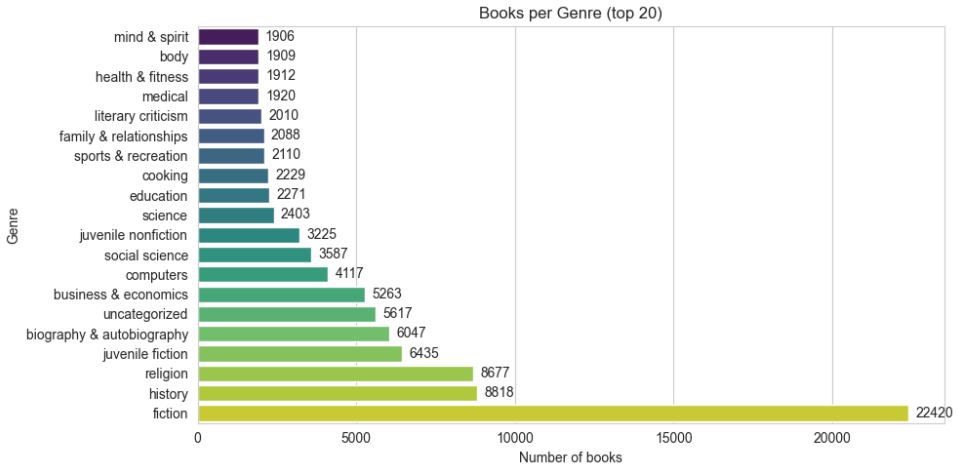


Figure 5: Top 20 Genres with the most books published

4 Machine Learning

Since the idea behind our project is to recommend books based on people's previous interests, we decided to compose a similarity score that reflects the main factors readers consider when choosing new books. During the exploratory data analysis (EDA) phase, we found that these text-based features were the most informative, as other metadata did not provide strong additional signals. Therefore, we mostly relied on existing natural language processing techniques in creating the similarity score and what we thought reflects the most important part in making a book like another.

In the real world, readers often rely on book descriptions and reviews to gage whether they might enjoy a book, and genres help narrow down choices to their preferred themes or styles. Descriptions and reviews sometimes even mention other popular books for comparison, making the specific words and phrases used in these texts highly valuable for recommendation. To capture these nuances, we applied TF-IDF vectorization, which transforms the text into numerical vectors that emphasize distinctive words such as themes, authorial style, and references to other works, while down-weighting common or less informative terms. This mirrors the way humans intuitively compare books by focusing on what makes each one unique.

However, raw TF-IDF vectors are typically very high-dimensional and can be noisy, which makes direct similarity calculations inefficient and less robust. To address this, we incorporated Truncated SVD, a dimensionality reduction technique that compresses the TF-IDF vectors into dense, low-dimensional embeddings. These embeddings retain the most important information for distinguishing books, and we further normalize them so that similarity calculations using dot products approximate cosine similarity which is a commonly used measure for comparing text vectors.

Our pipeline begins by normalizing titles and reviews and aggregating reviews per title. TF-IDF vectorization then converts each description and each aggregated review into high-dimensional sparse vectors that upweight words that are important for a particular book and down weight common words across the free-text data. Because raw TF-IDF is usually large and noisy, Truncated SVD compresses those matrices into dense, low-dimensional embeddings and each embedding is L2-normalized so dot products approximate cosine similarity.

When a user provides a list of favorite books, our system maps these titles to their corresponding vectors, averages the vectors to create a personalized preference profile, and compares this profile to all other books using fast matrix operations. The final similarity score is a weighted combination of description and review similarities, with configurable weights to reflect their relative importance. By default, the genre and description are emphasized more heavily, while reviews contribute a smaller but still meaningful component. We also add a genre-match bonus, so books that share genres with the favorites receive a boost in their scores.

To support our goal of promoting independent authors, the recommendations are split into indie and non-indie lists, allowing users to discover unique and lesser-known works that best match their interests. This approach ensures that our recommendations are both personalized and diverse, highlighting books that are similar in content and style to the user's favorites, while also giving visibility to indie authors who might otherwise be overlooked.

5 Website

To display our results, we built an interactive web application using StreamLit. It was inspired by the BoRiS (Book Recommendation System) GitHub project [2]

The application features a simple intuitive search-driven interface where the user searches for a book by title or author by typing it, then they can select 2 to 3 books, and click on a button to get recommendations. The search functionality employs fuzzy string matching to accommodate partial entries or misspellings. The recommendation models are created in the previous step and when the app is deployed for the first time. Clicking the "Get Recommendations" button computes the similarity scores using the pre-trained models and returns results in two sections: indie recommendations are displayed first to maximize visibility for independent authors, followed by mainstream titles. Each recommended book includes its cover image (when available), author, genre, average rating, and a match score representing similarity to the user's choices. Descriptions are also available in expandable sections, and users can access the book's Google Books page for more information. The application also provides a CSV export feature for users who want to save their recommendations and displays detailed statistics in a collapsible table.

6 Conclusion

Our indie book recommender successfully created a simple and functional system that helps users discover new underrated authors. The main components worked well, and the web app made the system accessible to non-technical users. However, not everything went as planned. The exploratory data analysis yielded fewer significant insights for our recommendation strategy than what we expected. Another limitation is that our data comes exclusively from an outdated amazon web scraping dataset, so recent indie releases and more niche platforms are very underrepresented. Initially,, we were planning on web scraping using the GoodReads API as well as some social media websites, but in recent years many of these sources have become private and inaccessible, which makes it more challenging to find the appropriate and recent data for our project. The indie classification is also not very inclusive and may misclassify some authors who, for example, are very established but are self published and may not have many reviews specifically on amazon. The recommendation algorithm is also content-based, and ignores collaborative signals that could capture reader preferences that are not evident in text alone. If we were to start over, we would collect more recent data from multiple sources beyond amazon and include additional indie author verification methods. Future improvements could also include collaborative filtering alongside our current approach, integrating real-time data feeds, implementing user accounts for more personalized recommendations, and even developing a mobile app for greater accessibility.

References

[1] M. Bakhet. Amazon books reviews. Kaggle, 2020.

[2] T. Blanchfield. Boris: Book recommendation system. GitHub Repository, 2024.