Project Report: IMDb Top 250 Movies Scraper
About the Project
Title: My IMDb Top 250 Movies Scraper Project
Date: March 14, 2025
Author: Neeraja Gude

This is my first try at web scraping! I wanted to get the top 250 movies from IMDb's website (https://www.imdb.com/chart/top/)—their titles, years, and ratings. I used Python to scrape the data, fix it up, and save it in CSV files.

Project Overview: IMDb Movie Data Processing
Below is a detailed and informative breakdown of the steps I followed to scrape, process, and clean movie data from IMDb using a Jupyter Notebook. Each step builds on the previous one to ensure a well-organized and reliable final data set.

Step 1: Webpage Scraping and Storing Raw Data
Objective: Collect initial movie data from the IMDb website.
Process:

- Utilized the `requests` library to fetch the IMDb webpage, including custom headers to mimic a browser and avoid being blocked.
- Employed `Beautiful Soup` to parse the HTML content and extract key information such as movie titles, release years, and ratings.
- Saved the raw, unprocessed data into a file named `imdb_raw_data.csv` for further analysis.

Outcome: A raw dataset containing all scraped movie details, ready for initial review.

Step 2: Generate Initial Data Sample
Objective: Verify the accuracy of the scraped data.
Process:

- Displayed the first 5 entries from `imdb_raw_data.csv` to inspect the structure and content (e.g., titles, years, ratings).
- Saved this sample to ensure a checkpoint for validation.

Outcome: A quick visual confirmation of the data's integrity, helping to identify any immediate issues in the scraping process.

Step 3: Process the Data
Objective: Convert raw data into a usable format.
Process:

- Transformed the release years and ratings from text to numerical values for easier analysis.
- Stored the processed data in a new file, `imdb_processed_data.csv`.

Outcome: A dataset with standardized numerical fields, improving compatibility for future calculations or sorting.

Step 4: Generate Processed Data Sample
Objective: Review the changes made during data processing.
Process:

- Displayed the first 5 entries from `imdb_processed_data.csv` to compare with the raw sample and confirm the numerical conversion.

Outcome: A clear view of the processed data, ensuring the transformations (e.g., years and ratings as numbers) were applied correctly.

Step 5: Clean the Data
Objective: Remove inaccuracies and irrelevant entries.
Process:

- Identified and removed rows with missing values (e.g., no rating or year) or outdated years (e.g., pre-1900 data).
- Saved the cleaned dataset to `imdb_cleaned_data.csv`.

Outcome: A refined dataset free of errors and irrelevant entries, enhancing data quality for final use.

Step 6: Prepare Final Dataset
Objective: Organize the data for practical application.
Process:

- Sorted the cleaned dataset by rating in descending order to highlight top-rated movies.
- Saved the final organized list to `imdb_final_data.csv`.

Outcome: A polished, sorted dataset ready for analysis, presentation, or further use, with the highest-rated movies easily accessible.

Summary:
This structured approach ensured that the IMDb movie data was systematically scraped, processed, cleaned, and finalized. Each step included validation points to maintain data integrity, resulting in a high-quality `imdb_final_data.csv` that can be used for insights or visualization.

How I Did It
I used these tools:
Python: To write the code.
requests: To get the webpage.
Beautiful Soup: To find movie info in HTML.
pandas: To make tables and save CSV files.
I ran everything in Jupyter Notebook, splitting it into cells with Markdown notes to explain each step.

Results
I got these files:
imdb_raw_data.csv: The movies I scraped.
imdb_raw_sample.csv: First 5 raw movies.
imdb_processed_data.csv: Fixed years and ratings.
imdb_processed_sample.csv: First 5 processed movies.
imdb_cleaned_data.csv: Cleaned data.
imdb_final_data.csv: Final sorted list.

Problems I Faced
Understanding Errors: Sometimes my code stopped with weird messages like "NoneType has no attribute 'get_text'." It took me a while to figure out that meant some HTML parts were missing, and I had to keep trying different ways to fix it.
Finding the Right Tags: The HTML classes were tricky to find. I had to guess and check a lot.

What I Learned

- How to scrape a webpage with requests and BeautifulSoup.
- How to use pandas to fix and clean data.

- That some websites hide data with JavaScript, which is hard for beginners like me.

What I'd Do Next
Add more info like movie directors or runtime if I can find them.

Conclusion
I'm really pleased with how this project turned out! I didn't get every movie I hoped for, but I still managed to scrape some data from IMDb and work with it. I learned a bunch about web scraping and cleaning data, and it feels like a great first step into coding projects like this!