It can be understood that the first part of the problem is web scrapping and second is close to sentiment analysis and metrics calculation.

So first i started with web scraping using given input file and proceeded with calculating the metrics given in the expected output file.

In web scrapping identifying the correct tag to extract the exact expected out

To generate the output for (`article\_metrics.csv`) uploaded as output.csv in Google Drive using the provided Python script, the approach can be summarized as follows:

1. Setup and Dependencies:

- Ensure necessary libraries (`pandas`, `requests`, `beautifulsoup4`, `textstat`, `nltk`) are installed.

- Import required modules (`pandas`, `requests`, `BeautifulSoup`, `os`, `csv`, `nltk`).

2. Data Input:

- Read the input file (`input.xlsx`) containing URLs from which article content needs to be extracted.

3. Article Extraction:

- Define a function `extract\_article\_text(url)` to:

- Fetch the HTML content of each URL using `requests`.

- Parse the HTML using `BeautifulSoup` to extract the main article content (typically within `<p>` tags).

- Clean the extracted text by removing unnecessary characters and formatting.

4. Text Processing:

- Define utility functions to:

- `clean\_text(text)`: Tokenize, remove stopwords and punctuation from the article text.

- `calculate\_derived\_variables(text)`: Compute positive and negative scores, polarity, and subjectivity based on predefined dictionaries (`positive\_words` and `negative\_words`).

- `calculate\_readability\_metrics(text)`: Calculate metrics such as average sentence length, percentage of complex words, Fog index, etc., using `textstat`.

- `count\_personal\_pronouns(text)`: Count occurrences of personal pronouns (`I`, `we`, `my`, `ours`, `us`).

- `calculate\_avg\_word\_length(text)`: Compute the average word length in the article.

5. Processing Multiple Articles:

- Iterate through each URL from the input file:

- Extract article content using `extract\_article\_text(url)`.

- Save each extracted article as a text file (`{URL\_ID}.txt`) in a specified directory (`extracted\_articles`).

6. Output Generation:

- After extracting and saving all articles:

- Process each saved text file to calculate the derived variables and readability metrics.

- Aggregate results into a list of dictionaries (`results`), each representing metrics for one article.

- Write the aggregated results to a CSV file (`article\_metrics.csv`) containing metrics like positive score, negative score, polarity score, etc.

7. Execution:

- The main function `process\_text\_files(directory\_path, output\_csv)` orchestrates the entire process:

- Reads text files from `directory\_path`.

- Computes metrics for each text file using defined functions.

- Writes the final metrics to `output\_csv` in CSV format.

The approach involves reading URLs from an Excel file, extracting article content, processing the text to derive sentiment and readability metrics, and finally saving these metrics in a structured CSV file. This automated pipeline facilitates efficient analysis of multiple articles, enabling insights into their content characteristics.