**Significance**

Crawling wine data provides valuable insights for market analysis, enabling businesses to understand consumer preferences, price trends, and popular wine varieties. It supports the development of personalized recommendation systems and enhances customer experiences by tailoring suggestions based on individual tastes. The data also serves as a foundation for academic research in fields like natural language processing and data analytics, uncovering patterns and correlations. Additionally, it aids in competitive analysis, supply chain optimization, and consumer education, empowering smarter decisionmaking for both businesses and individuals while fostering innovation in digital and AIdriven applications within the wine industry.

**Detailed Crawling Process**

Here’s a stepbystep explanation of the crawling process:

1. Define the Data Requirements

Target Data: Crawl relevant wine information from `https://www.wineenthusiast.com/`, such as name, description, rating, price, origin, type, and winery.

Scope: Focus on specific wine titles provided in the existing dataset (`wine\_titles` list).

Output Format: Append the newly crawled data to the existing CSV file.

2. Setup the Crawling Environment

Tools Used:

`requests`: To send HTTP requests and fetch webpage content.

`BeautifulSoup`: To parse and extract HTML data from the pages.

File Handling:

Use `pandas` to load the existing CSV file and append newly crawled data.

Save the final combined dataset back to the same file.

3. Define Target URLs and Request Method

Search Page: Use the base URL `https://www.wineenthusiast.com/search?q=` for querying wine titles.

Product Details Page: Extract product detail links from the search results and send secondary requests to gather complete information.

Request Headers: Use a `UserAgent` to mimic browser behavior and avoid detection as a bot.

4. Build the Crawling Logic

Step 1: Send Search Request

Construct the search URL by encoding the wine titles, replacing spaces with `%20`.

Use `requests.get` to retrieve the search result page’s HTML content.

Step 2: Extract Detail Links

Locate the product detail link from the search result page, typically an `<a>` tag containing the product title.

If a link is found, proceed to the detail page; otherwise, skip and log the missing result.

Step 3: Scrape Detail Page Data

Send another request to the detail page URL to fetch complete product information.

Use `BeautifulSoup` to extract the following fields:

Name (title): The product title.

Description: A brief description of the wine.

Rating (points): The consumer or expert rating.

Price: The wine’s price.

Origin (country): The country where the wine is produced.

Type (variety): The type of wine (e.g., Pinot Noir).

Winery: The name of the winery.

Step 4: Store Data

Append each crawled record to a list for later conversion into a DataFrame.

Combine the new records with the original dataset and save them to the file.

5. Run the Crawler

Request Frequency: Use `time.sleep(2)` to pause between requests, preventing server overload or blocking.

Error Handling: Log errors for failed requests or parsing issues to ensure the crawler continues running.

6. PostProcessing Data

Merge Data:

Use `pandas.concat` to merge the new records with the existing dataset.

Ensure data consistency and avoid duplicate entries.

Save Data:

Write the updated dataset back to the CSV file, overwriting the original.

7. Validate Results

After running the crawler, check the CSV file to ensure new data has been appended correctly and all fields are consistent. This process ensures the dataset is ready for further analysis or application.

**summary of the data cleaning process**

1. Remove Redundant Columns

The dataset contains a column named `Unnamed: 0`, which serves only as a row index and has no analytical value. This column was removed.

2. Handle Missing Values

Key Categorical Columns (`country`, `province`, `variety`): Missing values in these essential columns were filled with the most frequently occurring value (mode).

Other Categorical Columns (`designation`, `region\_1`, `region\_2`, `taster\_name`, `taster\_twitter\_handle`): These columns had significant missing data, so the missing values were replaced with "Unknown" to distinguish them clearly.

Price Column (`price`): Since `price` is a numerical variable, missing values were filled with the median to avoid skewing the distribution with extreme values.

3. Remove Duplicate Records

Duplicate rows were identified and removed to ensure that each record in the dataset was unique.

4. Validate Data Integrity

After cleaning, the dataset was checked to confirm that there were no missing values, and the distributions of key columns were analyzed to ensure that the cleaning process was effective.