Assignment No. 1

Problem Statement: Reading and writing different types of datasets.

Objective: The objective of this assignment is to familiarize ourselves with reading and writing different types of datasets including .txt, .csv, and .xml from the web and local disk storage. We will explore how to load these datasets into memory, process them, and save them to a specific location on the disk.

Prerequisite :

1. A Python environment set up with libraries like pandas, xml.etree.ElementTree, and requests (for web access).
2. Internet connection (for reading datasets from the web).
3. Text editor and basic knowledge of file handling in Python.

Theory :

In the realm of data analysis and data science, efficiently reading from and writing to various file formats is an essential skill for data processing and management. This assignment aims to help students become proficient in handling three common types of datasets: text files (.txt), comma-separated values files (.csv), and extensible markup language files (.xml). The task will guide students in performing file operations, including reading, modifying, and saving data, both from local disk storage and web sources.

Understanding Dataset Formats:

- **Text Files (.txt):**
    - Text files store data as unformatted plain text and are simple to manage. These files can contain unstructured or structured data with elements separated by spaces, tabs, or newlines.
    - They are commonly used for logs, configuration files, and basic datasets that don't require complex formatting.
- **Comma-Separated Values (.csv):**
    - CSV files store tabular data in a structured format where each line represents a row, and values are separated by commas or other delimiters. This format is compatible with most spreadsheet applications and databases, making it an industry-standard format for data exchange.
    - CSV is preferred for large-scale data processing due to its ease of use and simplicity.

- **Extensible Markup Language (.xml):**
    - XML is a flexible markup language used to encode documents and data in a structured and hierarchical way. It allows representation of complex data relationships using nested tags.
    - XML is widely used in web services, configuration files, and for data exchange between different systems.

Reading and Writing Different Dataset Formats:

Text Files (.txt):

- Python's open() function is used to handle text files. By using the built-in file methods, we can read, process, and write data efficiently.
- Example operations might include reading the file content, counting specific words or phrases, or filtering lines based on certain criteria.

CSV Files (.csv):

- CSV files can be handled easily with Python's Pandas library, which provides functions like pd.read_csv() for reading CSV files and DataFrame.to_csv() for writing them back to disk.
- Pandas enables easy manipulation of CSV data, including filtering, aggregating, and restructuring the dataset.

XML Files (.xml):

- XML data can be parsed using Python's xml.etree.ElementTree module. This library allows us to read XML files, access or modify specific elements, and save the changes back to the file.
- XML is especially useful for representing hierarchical data, and it requires careful navigation of the file structure to make modifications.

Steps for Loading Data into Memory:

Python Libraries and Tools:

- Pandas is a versatile library that simplifies reading and manipulating tabular data, especially CSV and XML files. It provides convenient functions to handle datasets with complex structures.
- open() is a basic function in Python used to work with simple text files for both reading and writing.

Implementation Algorithm:

1. Environment Setup:

- Ensure that the necessary libraries (Pandas, requests, and xml.etree.ElementTree) are installed. These libraries provide the functions to fetch, load, manipulate, and save datasets.

2. Reading and Processing a Text File (.txt):

- Open the text file using the open() function and load the content into memory.
- Process the content, such as performing a word count, removing empty lines, or extracting specific information.
- Write the processed content to a new text file.

3. Fetching and Processing a CSV File (.csv):

- Use the requests library to download a CSV file from a given URL.
- Load the CSV file into a Pandas DataFrame using pd.read_csv().
- Manipulate the data by filtering rows, applying statistical methods, or creating new columns.
- Save the modified DataFrame to a new CSV file using DataFrame.to_csv().

4. Reading and Modifying an XML File (.xml):

- Parse the XML file using the xml.etree.ElementTree module.
- Modify the XML structure by updating elements, attributes, or adding/removing nodes.
- Save the updated XML content back to a file using ElementTree.write().

Algorithm:

1. **Set Up Environment:**
    - Ensure required libraries (Pandas, Requests, xml.etree.ElementTree) are installed:
        - Install Pandas: `pip install pandas`
        - Install Requests: `pip install requests`
        - `xml.etree.ElementTree` is part of Python's standard library.

2. **Read and Process Text File (.txt):**

    a. Open the text file using Python's `open()` function and read its content.
    b. Process the content (e.g., count words, filter lines).
    c. Write the processed data to a new text file.

3. **Fetch and Process CSV File (.csv):**

    a. Use the `requests` library to download a CSV file from a URL.
    b. Load the CSV file into a Pandas DataFrame.
    c. Process the DataFrame (e.g., filter rows, aggregate data).
    d. Save the modified DataFrame to a new CSV file.

4. **Read and Modify XML File (.xml):**

    a. Parse the XML file using `xml.etree.ElementTree`.
    b. Modify the XML data as needed (e.g., update element values or attributes).
    c. Save the updated XML file back to disk.

References :

1.Kaggle

2.XML Processing in Python

3.Pandas Documentation

Conclusion:

This assignment will equip students with the ability to read, process, and write text, CSV, and XML data formats using Python. By mastering these operations, students will build a strong foundation in data management, which is crucial for data science workflows. The experience gained through this assignment will help students handle real-world datasets efficiently, across diverse applications such as web services, data interchange, and file-based data storage.