**Assignment No. 1**

**Problem Statement: R**eading 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 :**

 **Text Files (.txt)**:

* Plain text files typically contain unstructured or semi-structured data, which can be read and written using Python’s built-in file handling functions (open, read, write).
* Common methods for handling text files include read(), readlines(), write(), and writelines().

 **CSV Files (.csv)**:

* Comma Separated Values (CSV) is a structured format used for storing tabular data. Each row represents a record, and columns are separated by a delimiter, usually a comma.
* The pandas library provides efficient methods (read\_csv() and to\_csv()) for reading and writing CSV files.

 **XML Files (.xml)**:

* XML (eXtensible Markup Language) is a structured format for representing hierarchical data. It uses tags to define elements, making it suitable for representing nested data.
* Python's xml.etree.ElementTree module is often used for parsing, reading, and writing XML data.

 **Web Data Handling**:

* Data can be fetched from the web using the requests library, which supports HTTP requests. Once the data is retrieved, it can be processed and written to a file.

**Algorithm (if any to achieve the objective ):**

 **Reading from a File**: 1.1 **Text File (.txt)**:

* Use Python's built-in open() function to open the file.
* Use read() or readlines() to load data into memory.
* Close the file after reading.

1.2 **CSV File (.csv)**:

* Use the pandas.read\_csv() function to load the CSV file into a DataFrame.
* Optionally, specify parameters like delimiter, header, and index\_col.

1.3 **XML File (.xml)**:

* Use the xml.etree.ElementTree module to parse the XML data into an ElementTree object.
* Traverse the tree and extract the required data.

 **Fetching Data from the Web**:

* Use the requests.get() method to download data from a URL.
* Process the response (depending on the format, use appropriate methods to handle .txt, .csv, or .xml).

 **Writing Data to Disk**: 3.1 **Text File**:

* Open a file in write mode using open(file\_name, 'w').
* Use the write() method to save data into the file.

3.2 **CSV File**:

* Use the DataFrame.to\_csv() method from pandas to save the DataFrame to a CSV file.

3.3 **XML File**:

* Use the ElementTree.write() method to save XML data to a file.

**Conclusion** : This assignment demonstrates how to handle different types of datasets such as .txt, .csv, and .xml. It covers how to load data into memory, process it, and save it back to a specific location on disk. Additionally, it shows how to fetch and manage data from the web.