Assignment: XML Data Processing and Dashboard Development

Objective: The goal of this assignment is to process multiple XML files, convert them into a structured CSV format, apply feature engineering techniques, and visualize the data using a Streamlit dashboard.

Instructions:

1: XML Data Processing

1. Collect Data: You will be provided with multiple XML files containing structured data.
2. Parse XML Files: Extract relevant data fields from each XML file and convert them into a structured tabular format using Python. Use of data dictionary to apply meaningful column's structure.
3. Merge Data: Combine data from multiple XML files into a single Pandas DataFrame.
4. Export Data: Save the processed data as a CSV file.

2: Feature Engineering

1. Handle Missing Values: Identify and fill or drop missing data appropriately.
2. Data Transformation: Apply necessary transformations such as normalization, encoding categorical variables, or generating new meaningful columns.
3. Feature Selection: Select important features that contribute to insights and discard redundant ones.
4. Generate Derived Metrics: Create new derived metrics or aggregations that can enhance the analysis.

3: Data Visualization using Streamlit

1. Build a Streamlit Dashboard: Create an interactive dashboard to present insights from the processed data.
2. Include the Following Components:
    a. File uploader to allow users to upload XML files.
    b. Data preview section to display uploaded and processed data.
    c. Interactive visualizations such as:
        i. Bar charts
        ii. Line charts
        iii. Scatter plots
        iv. Aggregated metrics (e.g., mean, sum, count)
3. Filter and Search: Add filter options to allow users to explore data dynamically.

4. Deploy the App (Optional): Deploy the dashboard on Streamlit.

Additional Consideration for Evaluation:

- Implement error handling for missing or incorrect XML formats.
- Implement feature engineering techniques for additional signals.
- Implement EDA and insights in thoughtful reports/charts/graphs.
- Use of modular functions and classes with focus on readability and maintainability.
- Use caching in Streamlit to improve performance.
- Add user authentication to the dashboard.

Tools & Libraries Recommended:

- Pandas, Numpy, PyArrow (for data processing)
- BeautifulSoup (for XML parsing)
- Streamlit (for dashboard development)
- Matplotlib/Seaborn/Plotly (for data visualization)

Submission Guidelines:

- Upload your Python script (.py) or Jupyter Notebook (.ipynb) containing the code.
- Provide the CSV file generated after processing the XML files.
- Include README with instructions on how to run the Streamlit script.