**MLOps Implementation with Apache Airflow**

**Objective**

The primary objective is to implement Apache Airflow to automate the processes of data extraction, transformation, and version-controlled storage.

**Workflow**

Data Extraction

Sources: Data is extracted from Dawn and BBC websites.

Methodology:

* **Dawn Website**:

Scraping: The HTML content of the Dawn homepage is parsed using BeautifulSoup.

Data Extraction: Article links, titles, and descriptions are extracted by locating specific HTML elements, such as <a> tags with the class story\_\_link and their corresponding article descriptions.

* **BBC Website**:

Scraping: The BBC homepage is parsed using BeautifulSoup.

Data Extraction: Article links, titles, and descriptions are extracted by identifying specific data-testid attributes. Additionally, embedded JSON data is handled to extract article content.

**Data Transformation**

Text Cleaning:

* HTML Tag Removal: All HTML tags are stripped from the text.
* Special Character Removal: Unwanted characters are removed using regular expressions.
* White Space Normalization: Text is stripped and whitespace is normalized to ensure consistency.

Formatting:

* Lowercasing: All text is converted to lowercase to avoid case-sensitivity issues.
* Tokenization: Text is split into tokens to prepare for analysis (optional, based on requirements).

**Data Storage and Version Control**

Storage:

* Location: Processed data is stored on Google Drive.
* Format: Data is stored in CSV format to ensure compatibility with various analysis tools.

Version Control:

* DVC Initialization: Data Version Control (DVC) is initialized to manage data versions.
* Data Tracking: DVC is used to track versions of the data files.
* Remote Storage Configuration: DVC is configured to use Google Drive as the remote storage.
* Push Data: Data is pushed to the remote storage to ensure version control and backup.

**Apache Airflow DAG Development**

DAG Design:

Tasks: The DAG consists of three main tasks:

1. print\_working\_directory: Prints the current working directory.
2. extract\_data: Extracts data from the Dawn and BBC websites.
3. run\_dvc\_commands: Runs DVC commands to track and push data.

Dependencies:

* extract\_data runs after print\_working\_directory.
* run\_dvc\_commands runs after extract\_data.

Error Management:

The DAG handles task dependencies and includes error management to ensure smooth execution.

**Challenges**

* Website Structure Changes: Frequent changes in website structure required adaptable HTML parsing strategies.
* DVC and Remote Storage Configuration: Initial setup of DVC with Google Drive required careful configuration to ensure seamless data push and pull operations.

**Repository Maintenance**

GitHub Repository:

A well-documented GitHub repository is maintained with a detailed README.

The repository includes all scripts, documentation, and necessary files for easy setup and usage.

**Setup Instructions**

1. Clone the repository from GitHub.
2. Install required dependencies.
3. Set up Airflow and DVC as per the instructions in the README.

**Usage Instructions**

Run the Airflow DAG to start the ETL process.

Use DVC to manage and track data versions.

**Conclusion**

This implementation demonstrates the use of Apache Airflow to automate data extraction, transformation, and version-controlled storage, providing a robust and scalable solution for managing data workflows.