Assignment 1

Name: Snehal Laxmikant Yelwande

Class: TY A CSE(AI)

Roll no: 63

Batch P-2

PRN: 22311760

**Problem Statement:** Implement dataset versioning using DVC

Theory:

**Need of Data Versioning**

In machine learning and data-driven projects, datasets often evolve over time. Without proper versioning, it becomes difficult to track changes, ensure reproducibility, and collaborate effectively with team members. Traditional version control systems like Git are not optimized for handling large files, especially datasets and model artifacts.

To overcome these challenges, **Data Version Control (DVC)** provides a way to manage large data files, datasets, machine learning models, and experiments alongside code, while integrating seamlessly with Git. This assignment focuses on implementing dataset versioning using DVC to ensure reproducibility and efficient collaboration in ML projects.

**Data Version Control (DVC):**

DVC is an open-source tool designed to handle versioning of data, machine learning models, and pipelines. It extends Git’s version control capabilities to large files that cannot be stored directly in Git repositories.

**Key Concepts:**

1. **Dataset Versioning:**
   * Each dataset version is stored and tracked.
   * Allows reverting, branching, and comparing different dataset versions.
2. **Storage Backend:**
   * DVC uses remote storage (e.g., local folder, AWS S3, Google Drive, Azure, GCP) to store actual dataset files, while Git tracks only lightweight. dvc metafiles.
3. **Separation of Code & Data:**
   * Code remains in Git, while datasets/models are stored externally but linked via DVC.
4. **Reproducibility:**
   * Any collaborator can pull the exact dataset version using the associated .dvc file.
5. **Collaboration & Efficiency:**
   * Large files aren’t duplicated in Git.
   * Only changes (deltas) are stored, reducing storage cost and time.

**Workflow of DVC for Dataset Versioning:**

1. Initialize Git and DVC in the project.
2. Add dataset to DVC (dvc add).
3. Commit the dataset metadata file (.dvc).
4. Push dataset to remote storage (dvc push).
5. Share repository and .dvc file with collaborators.
6. Retrieve dataset version with dvc pull.

**Executions:**

1. **Created a Python script assignment1.py to download and save the Iris dataset as data/iris.csv.**

import pandas as pd

from sklearn.datasets import load\_iris

import os

os.makedirs("data", exist\_ok=True)

iris = load\_iris(as\_frame=True)

df = iris.frame

df.to\_csv("data/iris.csv", index=False)

print("Wrote data/iris.csv")

1. **Initialized Git and DVC:**

git init

dvc init

git add .dvc .dvcignore

git commit -m "Initialize Git and DVC"

1. **Added dataset to DVC tracking:**

dvc add data/iris.csv

git add data/iris.csv.dvc data/.gitignore

git commit -m "Track iris dataset with DVC"

1. **Configured local DVC remote storage:**

mkdir ../dvcstore

dvc remote add -d local\_remote ../dvcstore

git add .dvc/config

git commit -m "Add DVC remote"

1. **Pushed dataset to DVC remote:**

dvc push

**Output: Dataset successfully pushed to local DVC remote.**

***Link to the Github Repository***: [**MLOPS\_Assignments**](https://github.com/SnehalLY/MLOPS_Assignments)

**Conclusion**

Dataset versioning is crucial for ensuring reproducibility, transparency, and collaboration in machine learning workflows. Using DVC, datasets can be efficiently tracked, shared, and managed without bloating Git repositories. It bridges the gap between code versioning and data management, enabling teams to work on consistent and reliable datasets across multiple environments. By implementing dataset versioning with DVC, projects become more structured, maintainable, and future-proof.