Experiment 9: Data Version Control using DVC to Manage Large Datasets

Learning Objective: Students should be able to understand how to use DVC (Data Version Control) for managing large datasets, tracking data changes, and integrating it with Git for version control.

Tools: DVC, Git, Python, VS Code

Theory:

Data Version Control (DVC) is an open-source version control system tailored specifically for handling large datasets and machine learning models. It is designed to address the unique challenges posed by working with large, complex datasets and models, where traditional version control tools like Git are often inadequate. Just as Git revolutionized software development by enabling version control of code, DVC provides a similar approach for data and models, facilitating reproducibility, collaboration, and management throughout the lifecycle of a machine learning project.

In machine learning and data science workflows, managing datasets and models can be just as important as managing the code itself. As datasets grow larger and machine learning models become more complex, tracking changes to data, models, and the various stages of pipelines becomes increasingly difficult. DVC is designed to address these challenges by seamlessly integrating data management into the Git workflow.

Core Concepts of DVC:

1. **Data Versioning**:
   - Just as Git tracks changes to code, DVC tracks changes to datasets and machine learning models. DVC allows you to create, share, and collaborate on datasets efficiently while maintaining version history. This ensures that you can track different versions of your data and models across various stages of a project.
   - DVC does not store the actual large files (such as datasets) within the Git repository. Instead, it stores **metadata** and **pointers** to the actual data. This approach keeps the Git repository lightweight and efficient, as only metadata (such as file paths, hashes, and version information) is stored within the repository.

2. **Efficient Storage Management**:
   - DVC supports storing large datasets outside of Git, where the actual data files are stored in **external storage** systems. These can include:
     - **Cloud storage** (e.g., Amazon S3, Google Cloud Storage, Azure Blob Storage)
     - **Local storage** (e.g., hard drives, SSDs)
     - **Network storage** (e.g., network-attached storage (NAS), distributed file systems)
   - DVC then keeps track of these files and their versions using **pointers** in Git. This way, the project history remains lightweight while still allowing easy access to the original data.

3. **Reproducibility**:
   - One of the major advantages of using DVC is ensuring **reproducibility**. With DVC, you can define data dependencies and model versions within your Git repository. This makes it easier to reproduce machine learning experiments across different environments or by other team

members. For instance, if you're collaborating with a team, DVC ensures that every member can work with the exact same dataset, even across different versions of the project.

- DVC helps ensure that any machine learning model or dataset can be recreated and tracked as part of a pipeline. This is essential for maintaining consistency across different runs of experiments or when models need to be retrained.

Command & Output:

1. **Initialize DVC in a Git repository**

```
git init
dvc init
git commit -m "Initialize DVC"
```

2. **Add a dataset to DVC tracking**

```
dvc add dataset.csv
```

This command creates a `.dvc` file that tracks dataset changes.

3. **Commit dataset metadata to Git**

```
git add dataset.csv.dvc .gitignore
git commit -m "Track dataset with DVC"
```

4. **Push data to remote storage (e.g., Google Drive, AWS S3, local storage)**

```
dvc remote add myremote gdrive://your-folder-id
dvc push
```

5. **Retrieve dataset version from remote storage**

```
dvc pull
```

6. **Check the data pipeline status**

```
dvc status
```

7. **Remove a dataset from tracking**

```
dvc remove dataset.csv.dvc
git commit -m "Remove dataset from DVC tracking"
```

Conclusion:

Viva Questions:

1. What is DVC, and why is it useful?
2. How does DVC differ from Git?
3. What is a `.dvc` file, and what does it do?
4. How do you track a dataset using DVC?
5. How do you push data to a remote storage using DVC?

For Faculty Use:

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance/ Learning Attitude [20%]**	
Marks Obtained				