Getting Started with Data Version Control (DVC)

Why Versioning Data?

Working with Machine learning and data analysis often involve working with large datasets that change over time. Unlike code, managing datasets has unique problems:

- Changes to datasets are not easily trackable.
- Collaboration becomes difficult without a system to synchronize datasets across team members.
- Difficulties in Reproducibility when the dataset used for a specific model version is lost or modified.

Versioning data solves these issues by allowing:

- **Traceability**: Know exactly which dataset version corresponds to a specific model.
- **Reproducibility**: Recreate experiments with the same datasets used earlier.
- **Collaboration**: Share data efficiently within teams.

What is Data Version Control (DVC)?

DVC is an open-source tool designed to bring version control capabilities to data, making it work alongside traditional version control systems like Git.

- It **tracks datasets** and machine learning models in a Git-like fashion.
- DVC stores actual data in external storage (local, cloud, or remote), while saving lightweight pointers in the Git repository.
- It makes it easy for **data sharing, reproducibility, and efficient storage** by handling large files outside the repository.

How Does DVC Work?

DVC works in conjunction with Git to manage data:

- **Versioning**: Tracks changes in datasets by storing metadata in `.dvc` files.
- **Storage**: Data is stored in external storage backends (e.g., local directories, S3, Google Drive).
- **Pointers**: Git stores lightweight pointers (e.g., `.dvc` files) instead of the actual large datasets.
- **Automation**: DVC automates pulling and pushing data from/to the storage backend.

Using DVC

This guide walks you through setting up DVC in a project and managing your datasets.

---

1. Install DVC

First, We will need Git, as DVC relies on it. Then, download DVC from https://dvc.org/#get-started-dvc

Verify the installation:

```
dvc --version
```

2. Initialize DVC in Your Project

DVC needs to be initialized in your project directory. Start by creating a new project:

```
mkdir my-dvc-project
cd my-dvc-project
git init
dvc init
```

This creates a .dvc directory for configuration and adds it to your Git repository.

3. Add a Dataset to DVC

Place your dataset in the project directory (e.g., data/raw_dataset.csv) and add it to DVC:

```
dvc add data/raw_dataset.csv
```

This creates a .dvc file (e.g., raw_dataset.csv.dvc) that acts as a pointer to the data file. dvc then prompts you to add the generated .dvc and .gitignore files to git. Stage and commit the .dvc file to Git:

```
git add data/raw_dataset.csv.dvc data/.gitignore
git commit -m "Add raw dataset to DVC"
```

If you're working with a remote repo, and you do a git push, you'll notice the data file will not be pushed. This is because DVC handles data storage separately from the Git repository.

4. Configure Remote Storage

Set up remote storage to back up and share datasets.

Example 1, to use Amazon S3:

```
dvc remote add -d myremote s3://mybucket/path
```

Push the dataset to the remote storage:

```
dvc push
```

Example 2, to use Google Drive:

Install Google Drive Dependencies

```
pip install "dvc[gdrive]"
```

Configure Google Drive

1. Create a folder in your Google Drive (e.g., DVC_Storage).
2. Obtain the folder ID from the URL. For example:

```
https://drive.google.com/drive/folders/<FOLDER_ID>?<NOT_IMPORTANT_PARAMS>
```

3. Add the Google Drive remote to your DVC configuration:

```
dvc remote add -d myremote gdrive://<FOLDER_ID>
```

4. Push the dataset to the remote storage:

```
dvc push
```

5. Share the Project

Share the repository by pushing it to a Git remote:

```
git remote add origin <repository_url>
git push -u origin main
```

Other collaborators can clone the repository and pull the dataset:

```
git clone <repository_url>
dvc pull
```

6. Track Changes to the Dataset

If the dataset changes, DVC tracks the updates. For example, after modifying data/raw_dataset.csv:

```
dvc add data/raw_dataset.csv
git add data/.gitignore raw_dataset.csv.dvc
git commit -m "Update dataset"
dvc push
```

7. Clean Up Large Files

DVC optimizes storage by removing unnecessary large files:

```
dvc gc
```

This removes unused files from local and remote storage.