Proceso y comandos Pipeline DVC

1. **Preparación de DVC en el Proyecto**

```
# Inicializa DVC en el proyecto
dvc init
```

Este comando crea la configuración inicial de DVC en tu repositorio. Genera los archivos .dvc/config y .dvc/.gitignore, permitiendo que DVC maneje el versionado de datos y de los archivos grandes sin que estos se almacenen directamente en Git.

2. **Configurar Almacenamiento Remoto**

Para mantener una copia de seguridad de los datos y otros archivos generados, como los modelos entrenados, configuraremos un almacenamiento remoto en DVC. Esto puede estar en Google Drive, Amazon S3, o en un almacenamiento local.

```
# Agrega un almacenamiento remoto predeterminado en DVC (ejemplo con Google Drive)
dvc remote add -d myremote gdrive://<GDRIVE_FOLDER_ID>
```

Este comando configura myremote como el almacenamiento predeterminado donde DVC guardará los archivos de datos, modelos y resultados. El -d indica que será el almacenamiento remoto predeterminado. Si usas otro sistema de almacenamiento (como S3 o Azure), el formato de la URL cambiará.

3. **Versionar el Dataset con DVC**

Para rastrear el dataset principal (en este caso, data/data.csv), utilizamos el comando dvc add.

```
# Versiona el dataset principal
dvc add data/data.csv
```

Este comando crea un archivo .dvc (data/data.csv.dvc) y un registro en .gitignore para asegurar que Git no versionará el archivo directamente. En lugar de eso, DVC gestionará su versión en el almacenamiento remoto.

Después de añadir el dataset a DVC, realiza un commit en Git.

```
# Agrega el archivo de control .dvc y el .gitignore a Git
git add data/data.csv.dvc .gitignore
git commit -m "Agregar dataset principal al control de versiones de DVC"
```

4. **Definir el Pipeline en dvc.yaml**

El archivo dvc.yaml define las etapas del pipeline de forma estructurada. Cada etapa se configura con tres elementos principales:

- **cmd**: Comando a ejecutar para cada etapa (el script que realiza cada tarea).
- **deps**: Archivos de los cuales depende cada etapa.
- **outs**: Archivos o carpetas de salida generados por cada etapa.

Este archivo dvc.yaml configura tres etapas:

1. **Preprocess**: Ejecuta src/preprocess.py, toma como entrada el dataset data/data.csv y los parámetros en params.yaml, y genera los archivos preprocesados train_data.csv y test_data.csv.

2. **Train**: Ejecuta src/train.py, que entrena el modelo con train_data.csv, y guarda el modelo entrenado en models/best_model.pkl.

3. **Evaluate**: Ejecuta src/evaluate.py, que evalúa el modelo usando test_data.csv y genera un archivo de métricas metrics.json.

5. Ejecutar el Pipeline Completo con dvc repro

Una vez que el pipeline está definido en dvc.yaml, puedes ejecutarlo con el siguiente comando:

```
dvc repro
```

Este comando lee dvc.yaml y ejecuta las etapas en el orden definido. DVC comprueba si los archivos de entrada y de salida para cada etapa han cambiado desde la última ejecución:

- Si no ha habido cambios, DVC omite la etapa.
- Si los archivos han cambiado, DVC vuelve a ejecutar la etapa y actualiza el pipeline.

Este enfoque asegura que sólo las etapas que requieren actualización se vuelvan a ejecutar, optimizando el uso de recursos y tiempo.

6. Versionar y Guardar Resultados en Git y DVC

DVC utiliza Git para rastrear los archivos de configuración y para capturar el estado de los archivos en el pipeline (dvc.yaml, dvc.lock y .dvc). Para mantener la reproducibilidad, es importante guardar estos archivos en Git:

```
git add dvc.yaml dvc.lock metrics.json models/best_model.pkl
git commit -m "Pipeline de DVC completo con preprocesamiento, entrenamiento y evaluación"
```

Este comando realiza un commit de dvc.yaml, dvc.lock, metrics.json, y el modelo entrenado, models/best_model.pkl. Con esto, cualquier persona puede clonar el repositorio y ejecutar el pipeline para obtener los mismos resultados.

7. **Guardar Archivos en Almacenamiento Remoto de DVC**

   Después de ejecutar el pipeline y realizar un commit, necesitas empujar los archivos grandes, como el dataset y los modelos entrenados, al almacenamiento remoto configurado:

   ```
   dvc push
   ```

   Este comando envía los archivos .dvc y las salidas de cada etapa (train_data.csv, test_data.csv, best_model.pkl, etc.) al almacenamiento remoto. Esto permite que otros usuarios del proyecto recuperen los datos y resultados sin tener que almacenar todo en Git.