Версионирование данных и экспериментов с DVC

Ирина Степановна Трубчик

https://t.me/+PsC-JDrwrvsxNmVi

Лекция 3

Цели занятия

1. Освоить базовые концепции версионирования данных для ML
2. Научиться отслеживать зависимости и результаты экспериментов
3. Построить воспроизводимый ML pipeline с DVC

Опрос на старт-)

Какие проблемы с данными при повторном обучении моделей возникали у вас?

Опрос: Какие проблемы возникали с данными при переобучении моделей?

- Потеря версии
- Разные датасеты
- Неочевидные изменения
- Данных не сохраняли

Проголосовать

Что такое DVC

- **Data Version Control — "Git для данных и ML-артефактов"**
- **Интеграция с Git, хранение big data вне репозитория**
- **Контроль стадий, параметров, метрик, зависимостей**

proglib

Git для Data Science: контроль версий моделей и датасетов с помощью DVC

dvc-documentation
spbstu-applied-math/dvc-documentation

Базовый workflow

```bash
git init
dvc init
dvc add data/raw/data.csv
git add data/raw/data.csv.dvc .gitignore
git commit -m "Add raw data with DVC"
dvc remote add -d storage s3://bucket
dvc push
```

Примечание:

- DVC-файл содержит hash, размер, путь
- Интеграция в PR-процессы: проверка изменений данных
- Remote хранит сами файлы

dvc.yaml: конвейер стадий


```text
stages:
  preprocess:
    cmd: python preprocess.py
    deps:
      - data/raw/data.csv
      - preprocess.py
    outs:
      - data/processed/data.csv
  train:
    cmd: python train.py
    deps:
      - data/processed/data.csv
      - train.py
    outs:
      - model.pkl
    metrics:
      - metrics.json
```

Воспроизводимость

- ***Каждая версия данных и кода связана через DVC и Git***
- ***Любой результат можно повторить, получить точно такие же артефакты***
- ***DVC lock-файл фиксирует актуальные версии зависимостей***

!!! Попробуйте переключиться на любую версию:

```bash
git checkout <commit>
dvc checkout
```

DVC exp: эксперименты

- Эксперименты как отдельные ветки/типы прогонов: параметры, метрики, зависимости

```bash
dvc exp run -S lr=0.1
dvc exp show
dvc exp apply <exp_id>
```

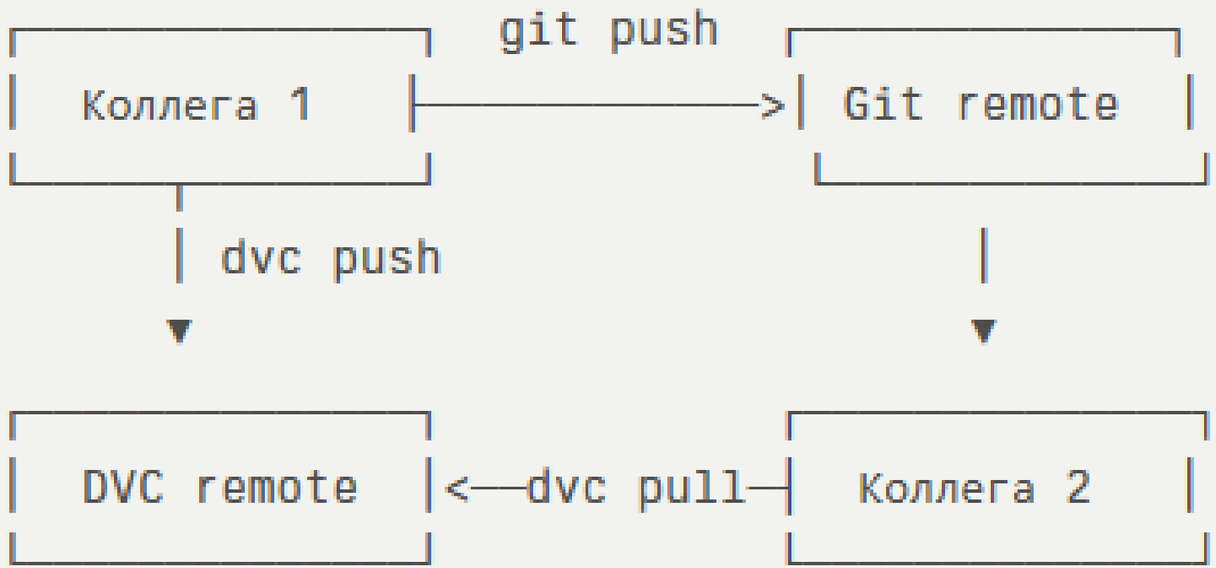
!!! сравните метрики и конфигурации экспериментов

DVC push/pull и удаленные хранилища

- DVC поддерживает множество remotes: S3, GCS, Azure, SSH, локальные
- Артефакты данных хранятся вне репо, только dvc-файлы в Git
- Совместная работа в команде: все видят структуру, версии, параметры

Графика процесса

text



Разница между push и pull для BigD

Критерий	Push	Pull
Инициатор	Поставщик данных «толкает» артефакты в удалённое хранилище.	Потребитель «вытягивает» данные из удалённого хранилища по метаданным.
Куда идут большие файлы	В DVC remote или LFS-сервер вместе с обновлением ссылок в Git.	С удалённого DVC/LFS хранилища на локальную машину по .dvc/.lock или LFS-указателям.
Необходимые права	Нужна запись в хранилище данных и право на публикацию изменений.	Достаточно прав чтения на хранилище и доступ к репозиторию.
Типовые команды	dvc push + git push (для метаданных и кода).	dvc pull + git pull (для данных и ссылок/кода).
Риск раздувания Git	Низкий: в Git хранятся только указатели/метаданные, не бинарники.	Низкий: данные забираются по указателям, не коммитятся в Git.

Автоматизация в CI/CD: data diff и метрики

- DVC metrics diff — сравнение метрик между коммитами или экспериментами

```bash
dvc metrics diff HEAD~1 HEAD
```

- Data diff — отслеживание изменений файла
- Gate-правила: не проходить PR, если качество падает

Валидация данных и мониторинг

1.Интеграция с Pandera/Great Expectations для проверки схемы

2.Хранение метаданных, автотесты данных в конвейере

Пример Pandera-теста в pipeline

```python
import pandera as pa
# Добавить на этапе preprocess
```

Use case: командный ML-проект

1.Сценарий: команда ведет единый репозиторий, DVC хранит данные, результаты, модели

2.PR-валидаторы проверяют изменения данных, запускают pipeline, сравнивают метрики

```python
import pandera as pa
# Добавить на этапе preprocess
```


CI/CD Pipeline
src
src
tests
config
docs
main.py
requiremens.txt
Сборка
Тестирование
Деплой
Релиз

Практикум

Задачи

- Создать свой data-репозиторий с DVC и remote
- Оформить dvc.yaml для feature pipeline
- Прогнать минимум два эксперимента с разными параметрами
- Сравнить результаты с помощью exp show и metrics diff

Контрольные вопросы

1. Для чего нужен dvc.lock?
2. Чем DVC отличается от обычного Git для ML?
3. Как проверить, что метрики не ухудшились?
4. Какие типы remote поддерживаются?
5. Как организовать reproducible ML workflow?

Материалы и ссылки

1. DVC Docs: https://dvc.org/doc
2. Quickstart: https://dvc.org/doc/start
3. DVC Experiments: https://dvc.org/doc/user-guide/experiment-management
4. DVC Metrics: https://dvc.org/doc/command-reference/metrics

Вопросы

Телеграм https://t.me/+PsC-JDrwrvsxNmVi

СКИФ (https://do.skif.donstu.ru/course/view.php?id=7508)