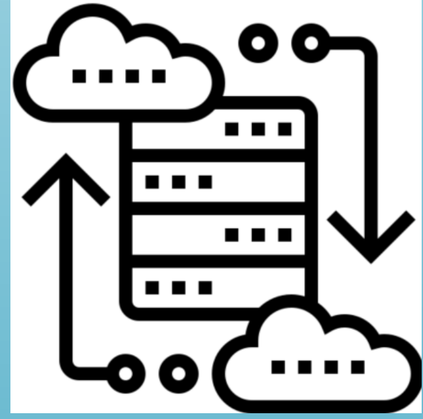


Управление данными и репозитории в ML

Ирина Степановна Трубчик

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

Лекция 2

О чем эта лекция

1. Почему данные — самая сложная часть ML
2. Типы данных и их особе
3. Системы версионирования данных
4. Хранилища признаков (Feature Stores)
5. Инфраструктура данных для ML
6. Российские аналоги инструментов

Проблема данных в ML

Основная проблема: в отличие от кода, данные:

- Имеют огромный объем (гигабайты → терабайты)
- Постоянно меняются и растут
- Имеют сложные зависимости и lineage
- Требуют специальной инфраструктуры для хранения

Проблема данных в ML

Пример: Модель компьютерного зрения требует 100 000 размеченных изображений по 1 МБ каждое = 100 ГБ данных.

Как версионировать такие объемы?

соотношение времени в ML-проектах

Сырые данные (Raw Data)	Обработанные данные (Processed Data)	Признаки (Features)
• Необработанные данные из источников • Пример: логи сервера, сырые изображения, транзакции • Особенности: большой объем, требует очистки	• Очищенные и преобразованные данные • Пример: нормализованные признаки, аугментированные изображения • Особенности: готовы для обучения	• Инженерные фичи, готовые для модели • Пример: эмбеддинги текста, статистики временных рядов • Особенности: требуют согласованности между обучением и инференсом

Типы данных в ML

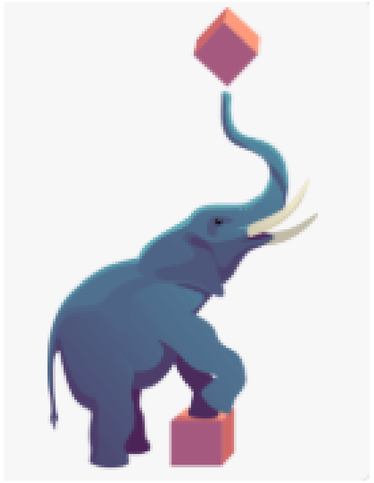
Версионирование данных

https://mlflow.org

Проблема: *Git не подходит для данных*

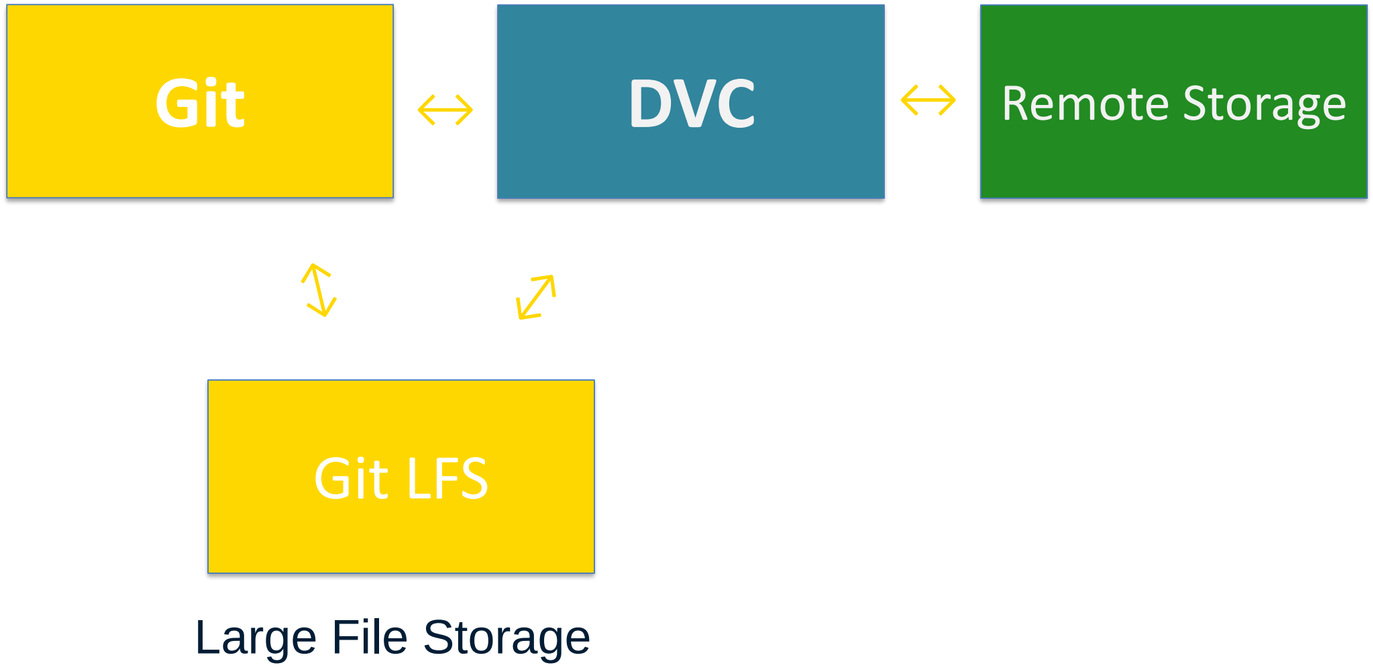
- ✓ Ограничения на размер файлов
- ✓ Нет эффективного хранения бинарных данных
- ✓ Медленные операции с большими файлами

Решение: Системы версионирования данных



https://pandera.readthedocs.io/en/stable/

Системы версионирования данных



Системы версионирования данных

Git LFS (Large File Storage)

- расширение для Git, которое оптимизирует работу с большими файлами, заменяя их в репозитории на небольшие файлы-указатели.
- Фактическое содержимое этих файлов хранится в отдельном удаленном хранилище, и подгружается только при необходимости (например, при клонировании или смене ветки).
- Это позволяет ускорить работу с Git, так как сам репозиторий остается небольшим, а репозиторий не раздувается от больших бинарных файлов.

Как работает Git LFS

1.Замена файлов указателями:
Вместо того чтобы хранить большие файлы (например, изображения, аудио, видео, САПР-файлы) в основном Git-репозитории, Git LFS создает небольшой файл-указатель.
2.Хранение больших файлов отдельно:
Сам большой файл хранится на специальном удаленном сервере или в выделенном LFS-хранилище.
3.Загрузка по запросу:
Когда вы клонируете репозиторий или переключаетесь на другую ветку, Git LFS скачивает нужные версии больших файлов из удаленного хранилища.
4.Прозрачная работа:
Для локального рабочего процесса большие файлы выглядят как обычные файлы, находящиеся прямо в репозитории, не требуя дополнительных действий от разработчика.

Преимущества Git LFS

Уменьшение размера репозитория:

Основной репозиторий остается компактным, что ускоряет клонирование и другие операции.

Ускорение работы:

Снижается время синхронизации и работы с репозиторием в целом.

Управление большими файлами:

Позволяет эффективно управлять бинарными файлами, с которыми обычный Git справляется плохо.

Когда использовать Git LFS

Git LFS полезен в проектах, где часто работают с большими бинарными файлами, такими как:

- Графические ресурсы для игр.
- Медиафайлы (аудио, видео).
- Файлы САПР-систем.
- Большие наборы данных.

https://docs.github.com/ru/repositories/working-with-files/managing-large-files/configuring-git-large-file-storage

Системы версионирования данных

DVC (Data Version Control)

- Работает поверх Git
- Хранит данные в удаленном хранилище (S3, Yandex Object Storage)
- Сохраняет в Git только метаданные и хэши

Pachyderm

- Специализированная платформа для данных
- Встроенная обработка данных через пайплайны
- Автоматическое версионирование

Архитектура DVC

Базовые команды

```bash
# Добавление данных под контроль версий
dvc add data/raw/dataset.csv

# Пушим данные в удаленное хранилище
dvc push

# Получаем данные по версии
dvc pull
```

Преимущества

- Прозрачная работа с данными любого размера
- Интеграция с ML-пайплайнами
- Поддержка различных бэкендов хранения

Хранилища признаков - Feature Stores

Проблема: *Рассогласование признаков между обучением и инференсом:*

- *Обучение: Признаки вычисляются на исторических данных*
- *Инференс: Признаки должны вычисляться в реальном времени*
- *Результат: Модель работает некорректно из-за разных данных*

Решение: Feature Store — централизованное хранилище признаков

Архитектура Feature Store

Процесс работы

Популярные Feature Stores

Feast (Open Source)	Tecton	Hopsworks
• Разработан Gojek • Поддержка Python, Go • Интеграция с крупными облачными провайдерами	• Управляемый сервис • Автоматическая обработка данных • Enterprise-функции	• Полноценная ML-платформа • Встроенный Feature Store • Поддержка on-premise развертывания

Российские аналоги

Для DVC

- **Yandex DataSphere** + Object Storage
- **SberCloud ML Platform** с встроенным версионированием
- **In-house решения** крупных банков (Тинькофф, Сбер)

Для Feature Stores

- **Yandex ClickHouse** как offline-хранилище
- **Tarantool** для online-хранилища
- **Кастомные решения** на основе Redis/PostgreSQL

Инфраструктура данных для ML

Источники данных

- Базы данных,
- API,
- стримы данных

Обработка данных

- Apache Spark,
- Apache Flink
- Airflow для оркестрации

Хранение данных

- Data Lake (сырые данные)
- Feature Store (признаки)
- Model Registry (модели)

Сервисы данных

- ML-пайплайны
- Сервисы инференса

Best Practices

Разделение данных

- Raw/Processed/Features
- Разные права доступа
- Отдельные хранилища

Безопасность данных

- Шифрование на rest и in transit
- Маскирование чувствительных данных
- Контроль доступа

Метаданные и Lineage

- Отслеживание происхождения данных
- Автоматическое документирование
- Воспроизводимость экспериментов

Рекомендательная система

Задача: Построить систему рекомендаций для маркетплейса

Решение:

1.Данные: История просмотров, покупок, кликов

2.Feature Store:

- Offline: ClickHouse (исторические данные)
- Online: Redis (актуальные данные пользователя)

3.Версионирование: DVC для датасетов и моделей

4.Результат: Персонализированные рекомендации в реальном времени

Инструменты для разных задач

Для небольших проектов	Для средних проектов	Для enterprise-проектов
• DVC + облачное хранилище • SQLite/PostgreSQL для признаков	• Feast + ClickHouse/PostgreSQL • Airflow для оркестрации	• Tecton/Hopsworks • Apache Spark + Kafka • Полный MLOps-стек

Итоги

- Данные — критически важный актив в ML
- Версионирование данных обязательно для воспроизводимости
- Feature Store решает проблему согласованности данных
- Выбор инструментов зависит от масштаба проекта

Вопросы

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

СКИФ