Автоматизация пайплайнов в ML

Ирина Степановна Трубчик

Лекция 6

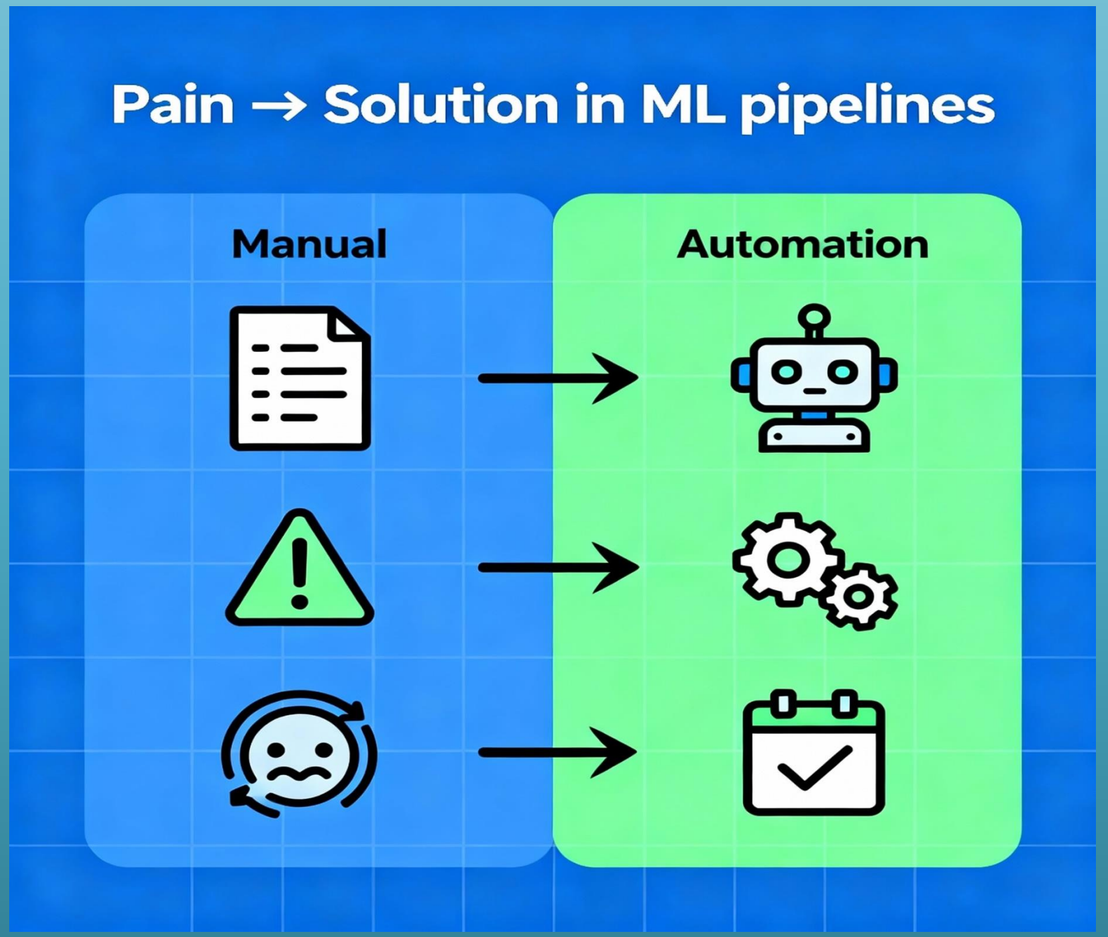
https://t.me/+PsC-JDrwrvsxNmVi

Цели занятия

1. Почему нужна автоматизация пайплайнов в ML
2. Современные системы: Apache Airflow, Prefect, Kubeflow Pipelines, DVC
3. Как связать ML пайплайн с CI/CD

Какая операция в ваших проектах самая рутинная и требует автоматизации?"
проголосовать

Почему нужна автоматизация ML-процессов



Автоматизация решает три ключевые проблемы:

- **Воспроизводимость — каждый запуск пайплайна гарантированно повторяет одну и ту же последовательность шагов с теми же параметрами.**
- **Снижение ошибок — ручные операции подвержены человеческому фактору (забыли запустить скрипт, перепутали файлы, использовали старую версию данных). Автоматизация устраняет эти риски.**
- **Скорость итераций — вместо часов на ручной запуск всех этапов, автоматизированный пайплайн запускается одной командой или по триггеру (коммит в Git, изменение данных, расписание).**

Дополнительные преимущества

- Легкая интеграция тестов (проверка данных, кода, метрик)
- Автоматическая генерация отчетов и уведомлений
- Возможность автоматического retraining при деградации качества модели
- Прозрачность для всей команды — любой член может посмотреть лог и понять, что происходило

Базовые этапы ML-пайплайна

Все эти этапы должны быть описаны в виде кода
и связаны зависимостями.
Изменение на любом этапе должно автоматически запускать
все последующие зависимые этапы

Какие из этих этапов в вашей практике чаще всего выполняются вручную и почему?"

Чем отличается автоматизация пайплайна ML от обычной автоматизации ETL

Аспект	ETL Pipeline	ML Pipeline
Детерминированность	Полностью детерминирован	Содержит стохастические компоненты
Проверка качества	Схемы данных, бизнес-правила	Метрики моделей, дрейф данных
Артефакты	Трансформированные данные	Данные + модели + метрики
Версионирование	Версии кода	Версии кода + данных + моделей
Мониторинг	Объем данных, время выполнения	+ качество предсказаний, дрейф
Rollback	Откат кода	Откат кода + модели + данных
Dependency	Только от данных	От данных + кода + гиперпараметров

Инструменты для оркестрации пайплайнов

сравнительная таблица

Что лучше для команды в малом стартапе (5-10 человек)?

Инструмент	преимущества	недостатки
Apache Airflow	✓ Богатая экосистема операторов (Python, Bash, Docker, Spark, и многие другие) ✓ Мощный UI для мониторинга и управления ✓ Поддержка SLA, алертов, повторных попыток ✓ Масштабируемость (поддержка Celery, Kubernetes executors) ✓ Активное сообщество и обширная документация	• Сложность настройки для новичков • Требует отдельной инфраструктуры (база данных, воркеры)
Prefect	✓ Чистый pythonic синтаксис, легкий старт ✓ Поддержка динамических workflow ✓ Встроенная интеграция с облачными сервисами ✓ Современная архитектура и UI ✓ Автоматическое кеширование и retry-логика	• Меньшее сообщество по сравнению с Airflow • Некоторые продвинутые функции доступны только в облачной версии

сравнительная таблица

Что лучше для команды в малом стартапе (5-10 человек)?

Инструмент	преимущества	недостатки
Kubeflow Pipelines	✓ Глубокая интеграция с Kubernetes ✓ Встроенная поддержка GPU, распределенного обучения ✓ Компонентный подход (каждый шаг — контейнер) ✓ Визуализация метрик и артефактов ✓ Идеально для облачных развертываний	• Требует Kubernetes кластера • Более крутая кривая обучения • Оверкилл для небольших проектов
DVC Pipelines	✓ Декларативный синтаксис (YAML) ✓ Тесная интеграция с Git ✓ Автоматическое кеширование и инкрементальные запуски ✓ Не требует отдельной инфраструктуры ✓ Идеально для воспроизводимых экспериментов	• Ограниченные возможности для сложных workflow • Нет built-in планировщика (нужен cron или CI/CD) • Меньше возможностей для мониторинга • Выбор инструмента зависит от: 1. Размера команды и проекта 2. Существующей инфраструктуры 3. Требований к масштабируемости 4. Опыта команды

Пример DAG-пайплайна на Airflow

```python
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta

# Функции для каждого этапа
def preprocess_data(**context):
    # Загрузка и обработка данных
    # Сохранение результатов
    pass

def train_model(**context):
    # Обучение модели
    # Логирование в MLflow
    pass

def evaluate_model(**context):
    # Валидация модели
    # Проверка метрик
    pass
```

```python
# Определение DAG
default_args = {
    'owner': 'mlops-team',
    'depends_on_past': False,
    'start_date': datetime(2025, 1, 1),
    'email_on_failure': True,
    'email_on_retry': False,
    'retries': 2,
    'retry_delay': timedelta(minutes=5),
}

with DAG(
    'ml_training_pipeline',
    default_args=default_args,
    description='ML pipeline for flight delay prediction',
    schedule_interval='@daily',  # Запуск каждый день
    catchup=False
) as dag:
```

DAG (Directed Acyclic Graph) — это граф задач с направленными связями без циклов. Каждая задача зависит от предыдущих и запускается только после их успешного выполнения.

Пример DAG-пайплайна на Airflow

```
# Определяем задачи
preprocess = PythonOperator(
    task_id='preprocess_data',
    python_callable=preprocess_data,
    provide_context=True
)

train = PythonOperator(
    task_id='train_model',
    python_callable=train_model,
    provide_context=True
)

evaluate = PythonOperator(
    task_id='evaluate_model',
    python_callable=evaluate_model,
    provide_context=True
)

register = BashOperator(
    task_id='register_model',
    bash_command='python src/register_model.py'
)

# Определяем зависимости
preprocess >> train >> evaluate >> register
```

Операторы — это обертки для разных типов задач:

PythonOperator — запуск Python-функции

BashOperator — выполнение bash-команды

DockerOperator — запуск контейнера

Зависимости (>>) — определяют порядок выполнения. preprocess >> train означает "train запустится только после успешного завершения preprocess".

Параметры запуска:

schedule_interval — частота запуска (cron-формат или предустановки типа @daily, @hourly)

retries — количество повторных попыток при сбое

retry_delay — пауза между попытками

email_on_failure — уведомления при ошибках

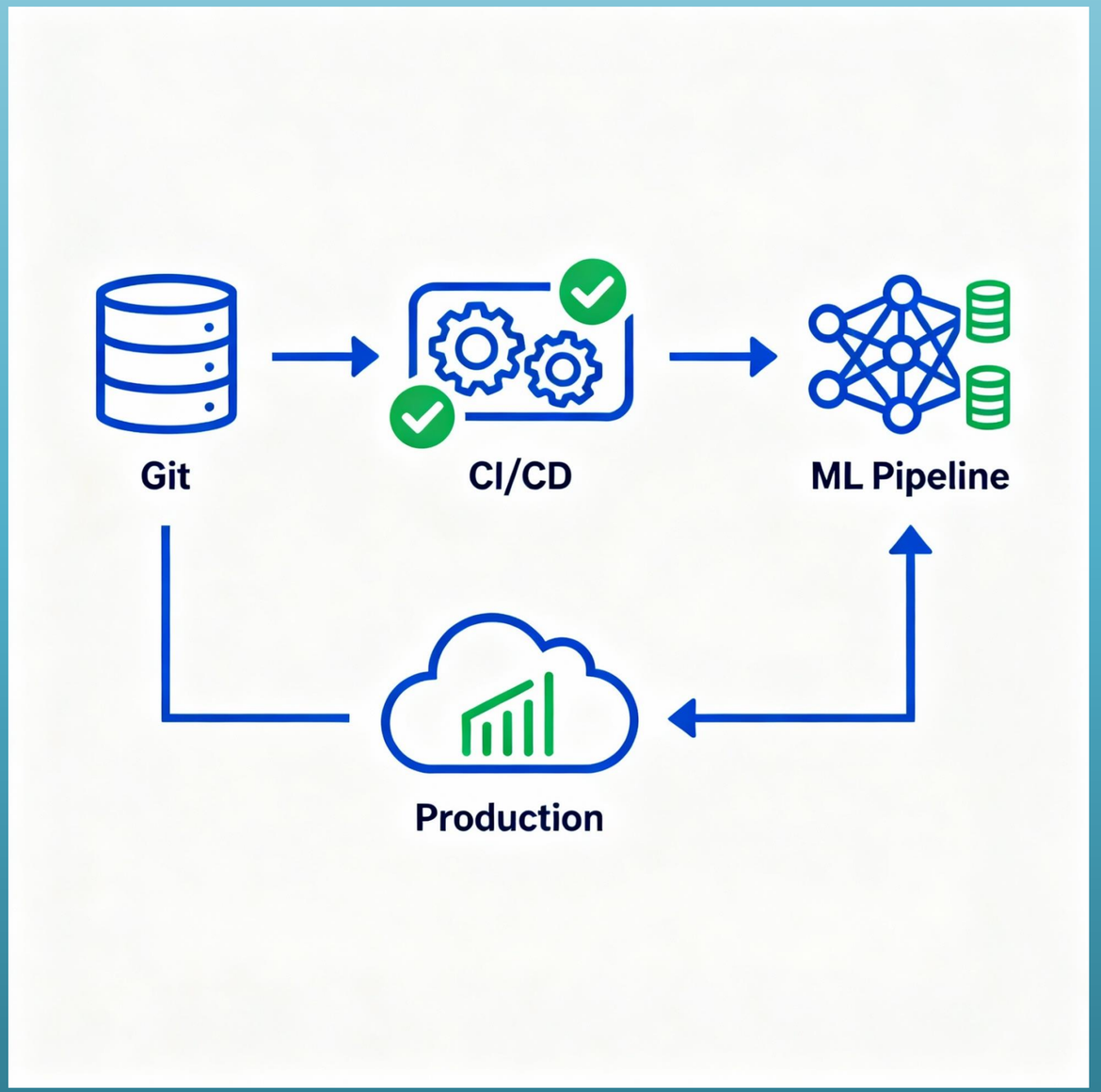
Где в этом пайплайне точка возврата при фейле?
Какие задачи можно было бы запустить параллельно для ускорения?

Преимущества такого подхода

- Визуализация всего пайплайна в UI
- Автоматические retry при временных сбоях
- Параллельное выполнение независимых задач
- Исторический лог всех запусков
- Возможность запуска отдельных задач вручную для отладки

Где в этом пайплайне точка возврата при фейле?
Какие задачи можно было бы запустить параллельно для ускорения?

Интеграция ML-пайплайнов с CI/CD



Причины интеграции

1. Автоматический запуск пайплайна при push/мерже в репозиторий
2. В связке: Checkpoints, gates на метриках (стоп, если качество плохое)
3. GitHub Actions/Jenkins/DVCCML проекция: build — test — pipeline — deploy
4. Логи, уведомления (Slack, email), автоматическое поднятие/откат моделей

Вопрос: Какие этапы стоит всегда делать "blocking" для перехода в прод?

Автоматизация retraining и мониторинга

- Мониторинг метрик качества и срабатывание перезапуска пайплайна
- Сценарии retrain: по времени (schedule), по дрейфу данных, по алерту метрик
- Автоматическая генерация отчетов (например, Evidently, MLflow
- Возможность уведомлений команде

Практикум: создаем свой пайплайн для автозапуска

- Опишите (на доске/в чате) свой ML-процесс в виде блок-схемы
- Подумайте, какие этапы можно описать как автоматические задачи
- Решите, какие проверки нужны для безопасного автоматического деплоя
- Для продвинутых: попробуйте реализовать простейший DVC pipeline или Airflow DAG локально

Контрольные вопросы

1. Что предотвращает автоматизация для ML-команды?

2. Кто должен иметь право запускать автоматический деплой и retraining?

3. Чем отличается автоматизация пайплайна ML от обычной автоматизации ETL?

Материалы и ссылки

1. https://airflow.apache.org/docs/
2. https://docs.dvc.org/doc/start/pipeline
3. https://mlflow.org/docs/latest/projects.html
4. https://docs.github.com/en/actions
5. https://evidentlyai.com/

Вопросы

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

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