Курс: MLOps (Machine Learning Operations)

Курс: 4-й курс бакалавриата
Объем: 14 лекций + 14 лабораторных
Цель курса: Освоение практик развертывания, мониторинга и поддержки ML-моделей в промышленной среде с использованием современных инструментов MLOps.

Лекция 1. Введение в MLOps

Теория:

- Что такое MLOps и зачем он нужен.
- Отличия DevOps и MLOps.
- Основные этапы жизненного цикла ML-модели.
  Лабораторная 1:
- Установка и настройка среды: Python, Conda, Jupyter, Git.
- Создание простого проекта ML.

Лекция 2. Управление данными и репозитории

Теория:

- Принципы работы с данными для ML.
- Хранилища данных (Data Lake, Data Warehouse).
- Инструменты версионирования данных: DVC, MLflow.
  Лабораторная 2:
- Версионирование датасета с помощью DVC.
- Интеграция с Git.

Лекция 3. Управление экспериментами

Теория:

- Понятие эксперимента в ML.
- Метрики, логирование, сравнение моделей.
- Инструменты: MLflow, Weights & Biases.
  Лабораторная 3:
- Создание ML эксперимента с MLflow.
- Сравнение моделей и сохранение метрик.

Лекция 4. Контейнеризация ML-приложений

Теория:

- Docker для ML: образы, контейнеры, Dockerfile.
- Почему важна контейнеризация для воспроизводимости.
  Лабораторная 4:
- Создание Docker-контейнера с простым ML-модулем.
- Запуск модели внутри контейнера.

Лекция 5. Оркестрация и CI/CD для ML

Теория:

- Основы CI/CD (Continuous Integration / Continuous Deployment).
- Отличия CI/CD для ML и классического DevOps.
- Инструменты: GitHub Actions, GitLab CI, Jenkins.
  Лабораторная 5:
- Настройка GitHub Actions для автоматического тестирования модели.

Лекция 6. Автоматизация пайплайнов ML

Теория:

- Пайплайны для подготовки данных, обучения, тестирования.
- Airflow, Kubeflow Pipelines.
  Лабораторная 6:
- Создание простого пайплайна Airflow для ETL и обучения модели.

Лекция 7. Развертывание ML-моделей

Теория:

- Подходы: REST API, gRPC, серверные фреймворки (FastAPI, Flask).
- Монолитное vs микросервисное развертывание.
  Лабораторная 7:
- Развертывание модели через FastAPI.
- Тестирование API с помощью Postman.

Лекция 8. Микросервисы и масштабирование

Теория:

- Контейнеризация + Kubernetes.
- Масштабирование моделей.
- Helm Charts, K8s Deployments.
  Лабораторная 8:
- Развертывание ML-контейнера в Minikube.
- Проверка масштабируемости.

Лекция 9. Мониторинг ML-систем

Теория:

- Метрики производительности моделей: latency, throughput, accuracy drift.
- Инструменты: Prometheus, Grafana.
  Лабораторная 9:
- Настройка мониторинга модели в Docker/K8s.
- Визуализация метрик в Grafana.

Лекция 10. Тестирование ML-моделей

Теория:

- Unit-тесты, интеграционные тесты для ML.
- Тестирование данных и моделей.
  Лабораторная 10:
- Написание pytest для проверки модели и пайплайна.

Лекция 11. Управление версиями моделей

Теория:

- Model Registry, управление версиями.
- Интеграция с CI/CD.
  Лабораторная 11:
- Регистрация моделей в MLflow Model Registry.
- Автоматическая подмена модели в пайплайне.

Лекция 12. Этические и юридические аспекты MLOps

Теория:

- Bias и fairness в ML.
- Конфиденциальность данных.
- Документирование моделей.
  Лабораторная 12:
- Анализ bias на небольшом датасете.
- Создание отчета о fairness модели.

Лекция 13. Feature Store и продвинутые инструменты

Теория:

- Feature Store: зачем и как.
- Инструменты: Feast, Tecton.
- Автоматизация feature engineering.
  Лабораторная 13:
- Создание простого feature store с Feast.
- Использование features в обучении модели.

Лекция 14. Итоги курса и практические кейсы

Теория:

- Примеры индустриальных MLOps кейсов (Netflix, Uber, Google).
- Основные тренды и будущие направления.
  Лабораторная 14:
- Развертывание end-to-end проекта с пайплайном, мониторингом и API.
- Презентация итогового проекта.

План курса "MLOps"

№	Тема лекции	Инструменты	Язык	Лабораторная работа
1	Введение в MLOps	Git, Conda, Jupyter	Python	Настройка окружения, создание простого ML-проекта
2	Управление данными и репозитории	DVC, Git	Python	Версионирование датасета с DVC и Git
3	Управление экспериментами	MLflow / Weights & Biases	Python	Логирование и сравнение моделей
4	Контейнеризация ML-приложений	Docker	Python	Создание Dockerfile и запуск модели в контейнере
5	CI/CD для ML	GitHub Actions / GitLab CI	Python + YAML	Настройка CI для тестирования ML-кода
6	Автоматизация пайплайнов	Apache Airflow	Python	Построение ETL + обучения модели в Airflow
7	Развертывание моделей	FastAPI, Flask, Postman	Python	Создание REST API для модели и тестирование
8	Микросервисы и масштабирование	Kubernetes (Minikube, kubectl)	YAML + Python	Деплой ML-контейнера в K8s, масштабирование
9	Мониторинг ML-систем	Prometheus, Grafana	YAML + Python	Настройка мониторинга метрик модели
10	Тестирование ML-моделей	pytest	Python	Написание unit- и интеграционных тестов
11	Управление версиями моделей	MLflow Model Registry	Python	Регистрация и управление версиями моделей
12	Этические аспекты ML	Pandas, Scikit-learn	Python	Анализ bias и fairness модели на датасете
13	Feature Store	Feast	Python	Создание feature store и использование признаков
14	Итоговое занятие	Все инструменты курса	Python + YAML	End-to-End проект: пайплайн, API, мониторинг

Итоговый демо-проект: "Предсказание задержки рейсов"

📌 Описание

Задача: построить ML-систему, которая предсказывает, задержится ли самолет более чем на 15 минут по расписанию.

Студенты должны:

1. Подготовить данные (ETL).
2. Обучить и зарегистрировать модель.
3. Запаковать модель в контейнер.
4. Настроить CI/CD для автоматизации.
5. Развернуть REST API для модели.
6. Организовать мониторинг качества и производительности.

🔧 Стек технологий

- **Данные:** открытый датасет авиарейсов (например US Flights 2018 или уменьшенный подмножество).
- **ML-библиотеки:** `scikit-learn`, `pandas`, `numpy`.
- **Управление данными:** `DVC`.
- **Эксперименты и модели:** `MLflow`.
- **Контейнеризация:** `Docker`.
- **CI/CD:** `GitHub Actions`.
- **Оркестрация:** `Airflow` или `Prefect`.
- **Развертывание:** `FastAPI` + `uvicorn`.
- **Мониторинг:** `Prometheus` + `Grafana`.
- **Feature Store:** `Feast`.

Структура проекта

```
mlops-flight-delay/
│
├── data/                # Данные (версируются через DVC)
├── src/
│   ├── preprocess.py    # Очистка и подготовка данных
│   ├── train.py         # Обучение модели
│   ├── predict.py       # Скрипт для предсказаний
│   ├── api.py           # REST API (FastAPI)
│   └── utils.py
│
├── models/              # Сохраненные модели (MLflow)
├── airflow/             # DAG для пайплайнов
├── docker/              # Dockerfile
├── tests/               # pytest тесты
├── .github/workflows/   # CI/CD
└── README.md
```

🚀 Этапы реализации

◆ Этап 1. Подготовка данных

- Очистить данные, выбрать признаки (например, день недели, месяц, авиакомпания, время вылета).
- Версионировать датасет с **DVC**.

◆ Этап 2. Обучение и управление экспериментами

- Обучить простую модель (например, `RandomForestClassifier`).
- Логировать параметры и метрики в **MLflow**.

◆ Этап 3. Контейнеризация

- Создать `Dockerfile` для запуска API.
- Запустить модель в контейнере.

◆ Этап 4. CI/CD

- Настроить **GitHub Actions** для:
  - Автоматического запуска тестов.
  - Обновления контейнера.

◆ Этап 5. Автоматизация пайплайна

- В **Airflow** настроить DAG:
  - Загрузка данных → Предобработка → Обучение модели → Регистрация в MLflow.

◆ Этап 6. Развертывание API

- Сделать REST API через **FastAPI**.
- Endpoint `/predict` принимает JSON с параметрами рейса и возвращает вероятность задержки.

◆ Этап 7. Мониторинг

- Настроить экспорт метрик (latency, количество запросов, accuracy drift) в **Prometheus**.
- Отображать в **Grafana**.

◆ Этап 8. Feature Store

- Вынести признаки (например, день недели, авиакомпания, сезон) в **Feast**.
- Подключить их к пайплайну обучения.

🏆 Результат

- End-to-End ML-система: от данных до продакшен API.
- Автоматизация с CI/CD и пайплайнами.
- Мониторинг и управление моделями.
- Готовый **мини-кейс для портфолио студента**.