



Train Delay Prediction

End-to-End MLOps Pipeline

КОТОВ Д.А.

ЗЫБИН М.А.

АРТЕМЕНКО А.В.

Проблема

Задержки поездов негативно влияют на пассажиров и логистику

Цель

Предсказать, какая будет задержка

Тип задачи

Sequential Prediction

Ключевые компоненты

DVC: Для версионирования данных и пайплайнов.

API: FastAPI с документацией Swagger.

CI/CD: GitHub Actions, Docker.

TensorFlow/Keras: Обучение моделей.

Airflow: Для оркестрации задач в пайплайне.

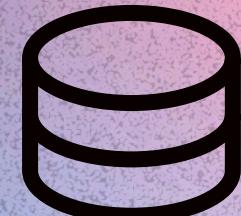
MLflow: Для логирования экспериментов и управления моделями.

Архитектура

```
└── airflow/
    └── data/
        ├── raw/
        ├── processed/
        ├── split/
        └── drift/
    └── .dvc
    └── .github/workflows/
        └── ci.yml
    └── mlflow/
    └── models/
    └── monitoring/
    └── reports/
    └── src/
        ├── preprocess.py
        ├── api/
        ├── train.py
        ├── evaluate.py
        └── wrapper_classes/
    └── tests/
        ├── test_api.py
        └── test_preprocess.py
    └── Dockerfile
    └── docker-compose.yml
    └── dvc.yaml
    └── environment.yaml
    └── pyproject.toml
    └── requirements.txt
    └── params.yaml
```

DAG'и для оркестрации процессов машинного обучения
Данные, отслеживаемые через DVC
Исходные данные
Обработанные данные
Разделённые данные
Данные для мониторинга дрейфа
Хранилище данных для DVC
CI/CD с GitHub Actions
Конфигурация CI (для тестирования и деплоя)
Логирование экспериментов с MLflow
Сохранённые модели
Мониторинг с Prometheus и Grafana
EDA-отчёты и отчёты по меткам
Основные скрипты проекта (обучение, предсказания, обработка)
Скрипт для предобработки данных
FastAPI сервис для предсказания задержек
Скрипт для обучения модели
Скрипт для оценки модели
Классы для обработки данных и предсказаний
Юнит-тесты и проверки
Тестирование API
Тестирование предобработки

Конфигурация DVC для пайплайнов
Совместимость с Conda
Зависимости проекта
Зависимости без UV
Параметры конфигурации проекта



данные

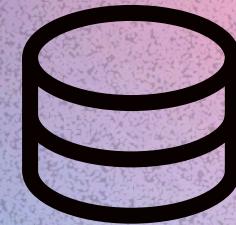
Источник данных - Kaggle: Train Delays Dataset

Признаки:

- Дата и время отправления.
- Идентификатор поезда, маршрут, перевозчик и время прибытия.
- Задержка в минутах.

Пример исходных данных:

id	datetime	carrier	connection	arrival	delay
ID001	2024-05-10 08:45:00	PKP Intercity	Wrocław Główny - Bydgoszcz Główna	21:48	5
ID002	2024-05-10 09:15:00	PKP Intercity	Warszawa - Kraków	10:20	0



Данные

Предобработанный датасет

Что мы изменили:

1. Переименовали столбец **date** в **departure_date**.
2. Создали новые признаки:
 - **year, month, day, dayofweek, hour** — на основе времени.
 - **season, is_weekend, is_holiday** — на основе даты и праздничных дней.
3. Циклическое кодирование часа (с использованием **hour_sin** и **hour_cos**).
4. Очистка данных: Пропущенные значения были заменены средними, строки с пропущенными важными данными были удалены.
5. Удалили лишние признаки: Например, **id_bracket** и **id_text**.

Пример предобработанных данных:

carrier	connection	name	id_main	arrival	delay	year	month	day	dayofweek	season	is_weekend	is_holiday	hour_sin	hour_cos
PKP Intercity	Wrocław Główny - Bydgoszcz Główna	Bydgoszcz Główna	ID001	21:48	5	2024	5	10	4	2	True	False	0.5878	0.8090
PKP Intercity	Warszawa - Kraków	Kraków	ID002	10:20	0	2024	5	10	5	2	False	False	0.9063	0.4226

Feature Store

Типы признаков:

Категория	Признаки	Тип данных
Route Features	avg_delay_7d, on_time_pct, volume	Batch (ежедневно)
Airline Features	airline_avg_delay_30d, reliability_score	Batch (ежедневно)
Temporal Features	hour, day_of_week, is_peak, season	On-demand
Weather Features	temperature, humidity, wind_speed	Streaming

Feature Store (Feast)

Offline (для обучения):

- Используется BigQuery для хранения признаков, версионирования и выполнения точных расчетов с учётом временной синхронизации (*point-in-time correctness*).

Online (для инференса):

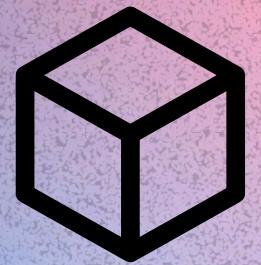
- Используется Redis для хранения признаков, которые необходимы для реального времени с задержкой ~50ms.

Версионирование:

- Все признаки версионируются, и история изменений отслеживается. Это гарантирует, что модели используют только актуальные данные.

Синхронизация:

- Признаки обновляются как в batch, так и в streaming режиме для обеспечения консистентности данных.



Выбор модели:

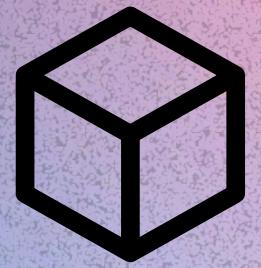
Для прогнозирования задержек использовалась модель на основе нейронных сетей (Keras, TensorFlow).

Метрики:

1. RMSE (Корень из средней квадратичной ошибки)
2. MAE (Средняя абсолютная ошибка)
3. R² (Коэффициент детерминации)

Визуализация результатов:

1. True vs Predicted (Истинные vs Предсказанные значения)
2. Residuals vs Predicted (Остатки vs Предсказания):



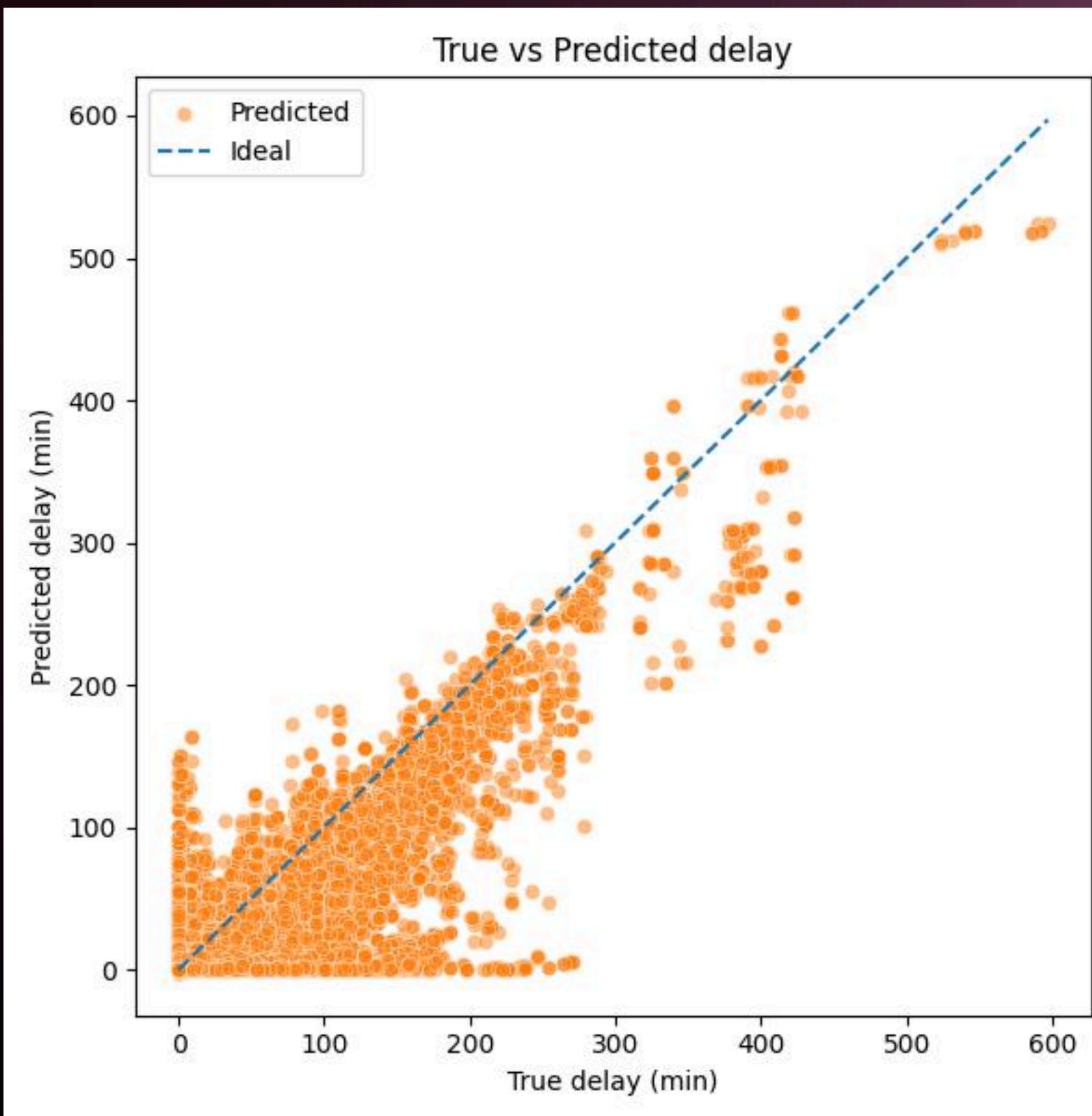
Оценка модели

Метрики

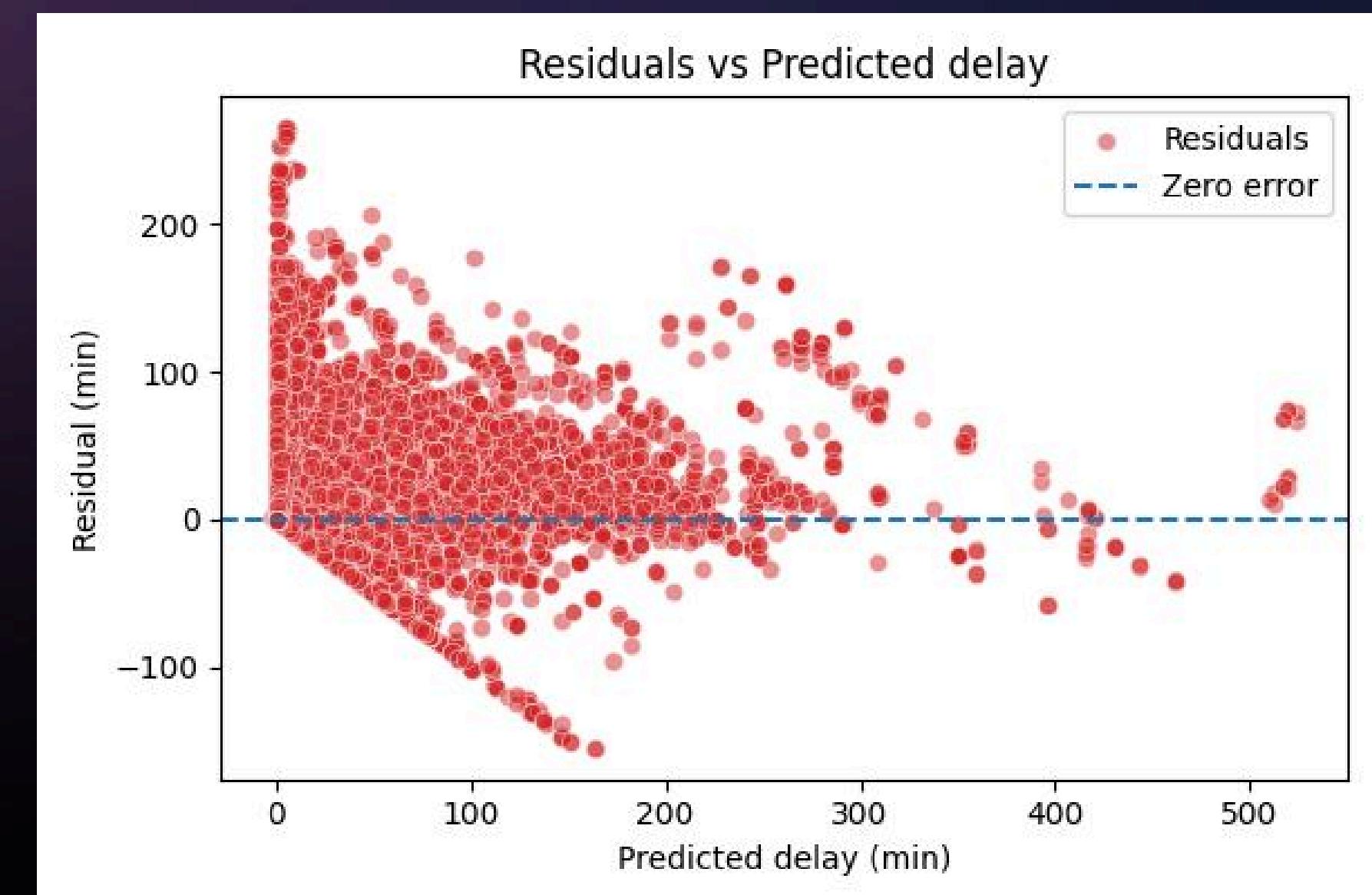
2.17
MAE:

9.15
RMSE:

0.60
 R^2



Графики



CI/CD Pipeline (GitHub Actions)

Commit & Hooks

Testing

DVC Repro

Docker build and push

Deployment

Линтинг и форматирование кода.
Используемые инструменты: `black`, `flake8`, `isort`, `mypy`.

Запуск юнит-тестов через `pytest`.
Покрытие тестами: $\geq 85\%$.
Включает тесты API через `TestClient` в `FastAPI` и проверки препроцессинга данных.

Воспроизведение пайплайна обработки данных с помощью DVC.
Используемые команды: `dvc repro`, отслеживание изменений данных и моделей.

Сборка Docker-образа приложения и загрузка его в Docker Hub Registry.
Используемые инструменты: `Dockerfile`, `docker-compose.yml`.

Тестовое развертывание в Kubernetes (с использованием KIND).
`Canary rollout` на `main branch` ($10\% \rightarrow 100\%$) с автоматическим откатом при ошибках.

Качество кода и мониторинг

Качество кода

Проверка линтинга и форматирования через flake8, black, isort, труу

Тестирование

Включает юнит-тесты для API и препроцессинга через pytest

Безопасность

Возможно использование инструмента для сканирования зависимостей (например, safety)

Качество кода и мониторинг

Основной эндпоинт:
/predict

POST /predict Predict

Parameters

No parameters

Request body required

Example Value | Schema

```
{  
    "carrier": "string",  
    "connection": "string",  
    "name": "string",  
    "id_main": "string",  
    "arrival": "string",  
    "year": 0,  
    "month": 0,  
    "day": 0,  
    "dayofweek": 0,  
    "season": 0,  
    "is_weekend": true,  
    "is_holiday": true,  
    "hour": 0  
}
```

Train Delay Prediction API 0.1.0 OAS 3.1 /openapi.json

default

GET / Root

POST /predict Predict

Schemas

HTTPValidationError > Expand all object

PredictRequest > Expand all object

ValidationError > Expand all object

Проблемы и вызовы

Данные и признаки:

- 1. Низкое качество исходных данных:** Отсутствие информации о скрытых факторах (например, ДТП, погодные условия).
- 2. Высокая динамика данных:** Необходимость регулярного обновления модели, чтобы она оставалась актуальной в условиях изменяющихся условий.

MLOps интеграция:

- 1. Сложность связи DVC, Feast, MLflow в едином пайплайне:** Трудности интеграции различных инструментов для обеспечения воспроизводимости и масштабируемости.
- 2. Настройка тестового деплоя в Kubernetes KIND:** Необходимость развертывания и настройки среды для тестирования и валидации модели в Kubernetes.
- 3. Обеспечение низкой задержки API при инференсе из Feast:** Потребность в обеспечении быстрых ответов от API для реального времени.

Тестирование и контроль качества кода

API Testing:

1. Pytest: Используется для модульного тестирования API. Пример: Проверка правильности возвращаемых данных по прогнозу задержки по API /predict.
2. Мокирование: Для быстрого тестирования логики API без обращения к внешним сервисам, например, с мокированием данных из Feast и MLflow.

Pre-commit Hooks:

1. Форматирование: Black (автоматическая обработка кода перед коммитом).
2. Линтинг: Flake8 (проверка стиля кода на соответствие PEP-8).
3. Проверка синтаксиса: Проверка файлов YAML, устранение конечных пробелов, проверка больших файлов.

Дорожная карта

Краткосрочные цели (1-2 месяца):

1. Внедрение дополнительных признаков для улучшения точности прогнозирования задержек (например, использование внешних данных о погоде).
2. Реализация активного обучения для повышения точности модели на сложных примерах.
3. Добавление интерпретируемости модели (например, с использованием SHAP-значений для предсказаний API).
4. Расширение функционала API для использования другими железнодорожными операторами.
5. Внедрение системы мониторинга с использованием Prometheus и Grafana для отслеживания производительности модели и мониторинга drift.

Среднесрочные цели (3-6 месяцев):

1. Разработка мульти-модальной ансамблевой модели (добавление LightGBM, CatBoost для улучшения точности).
2. Реализация механизмов адаптации модели в реальном времени (обработка дрейфа концепции).

Долгосрочные цели (6-12 месяцев):

1. Разработка системы рекомендаций для предложений альтернативных маршрутов в случае задержек.
2. Внедрение федеративного обучения для совместного использования данных от нескольких операторов.
3. Предсказания статуса движения поездов в реальном времени и интеграция с мобильным приложением.

Резюме проекта

Полный цикл MLOps:

От сырых данных до рабочего API, покрывающего все этапы жизненного цикла модели.

Воспроизводимость:

Использование DVC и MLflow гарантирует версионирование данных, кода и моделей.

Автоматизация:

CI/CD пайплайн обеспечивает быструю сборку, тестирование и деплой приложения в Kubernetes.

Команда

Котов Дэвид

Git, CleanCode, Feature Engineering
Ответственность за управление версиями (**Git**),
обеспечение чистоты кода (**CleanCode**),
создание и подготовку признаков для модели.

Зыбин Михаил:

ML, Training, AI
Ответственность за машинное обучение,
тренировка моделей, применение алгоритмов
ИИ для решения задач.

Артеменко Алексей

Датасеты, Обработка данных, Frontend
Ответственность за подготовку и обработку
данных, создание качественного датасета для
обучения моделей, создание интерфейса
взаимодействия с данными.