Предсказание цен на автомобили🚗

Full MLOps Pipeline

Выполнили:

Турсунов Р. А.

Савельева Е.К.

Жмаев В. А.

Структура и архитектура проекта

Технологический стек

- DVC: версионирование данных и пайплайнов.
- Feast: feature Store для хранения признаков.
- Модель: XGBoost с регистрацией в MLflow.
- API: FastAPI с документацией Swagger.
- CI/CD: GitHub Actions, Docker, Kubernetes KIND.
- Monitoring: Prometheus и Grafana.

Жмаев: настройка Airflow, Docker Compose, структура репо.
Савельева: создание Dockerfile и реализация задач обучения/препроцессинга в пайплайне.
Турсунов: разработка логики API (api.py) и интеграция этапов оценки модели.

Архитектура:

```
├── airflow          (Airflow DAG'и и конфигурация)
├── data
│   ├── features     (Сформированные признаки)
│   └── drift        (Отчёты дрейфа)
├── feature_repo     (Определение признаков в Feast)
├── k8s              (Kubernetes манифесты)
├── src (API, обучение, препроцессинг)
│   ├── api.py
│   ├── preprocess.py
│   ├── train.py
│   ├── train_best_model.py
│   ├── evaluate.py
│   ├── drift_check.py
│   └── tools
├── tests
├── Dockerfile
├── docker-compose.yml
├── dvc.yaml (ML Pipeline)
├── params.yaml
├── prometheus.yml
├── requirements.txt
└── README.md
```

Данные и обработка признаков

Савельева: реализация этапов preprocess (очистка, нормализация и тд).

Пайплайн обработки

- Очистка: удаление дубликатов и выбросов.
- Нормализация: приведение численных данных к единой шкале.
- Кодирование: One-hot и Label encoding для категорий.

Целевая переменная и фичи

- Target: цена автомобиля.
- Features: пробег, владельцы, топливо, мощность, регион и др.
- Хранение: Feast Feature Store обеспечивает доступность фичей для обучения и инференса.

Источник данных: Auto.ru

Объем данных:	432,227 строк о продаже автомобилей
Технические характеристики:	год выпуска, пробег, объем двигателя, топливо, тип трансмиссии, мощность (л.с.)
Рыночные признаки:	марка, модель, регион, цена (**целевая переменная**)
Исторические данные:	количество владельцев
Train/Test:	80%/ 20%
Стратегия разделения:	случайное с сохранением распределения марок/моделей.

Feature store: Feast

Консистентность признаков

Feast используется для унификации признаков между этапами обучения и инференса, исключая Feature/Serving Skew.

- Offline Store: хранение исторического набора данных для обучения XGBoost.
- Online Store: обеспечение низкой задержки для API-запросов предсказания.
- Регистрация: хранение метаданных о признаках (версии, источники).

Признаки (всего 17):

Категория	Примеры
Technical features	year, engine, power, transmission
Market features	region, model, mark
Historical features	owners

Жмаев: инициализация репозитория Feast и конфигурация хранилища.
Савельева: формирование Feature Views, оффлайн признаков и выполнение материализации.
Турсунов: интеграция Feast в процесс обучения и проверка воспроизводимости.

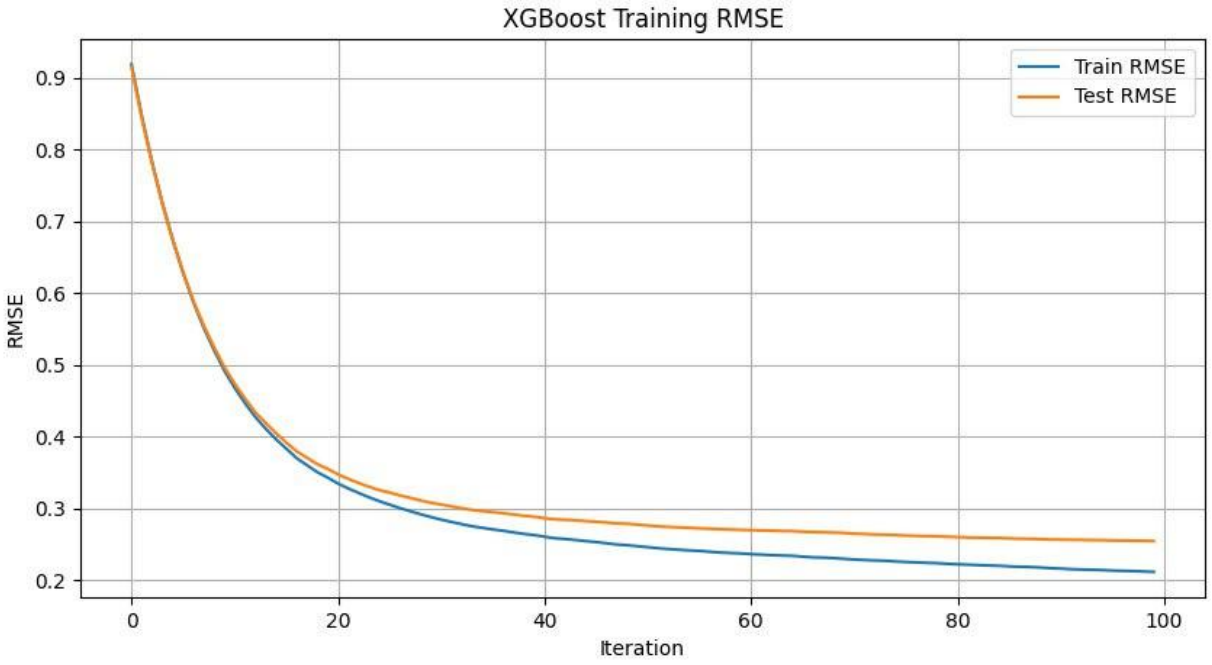
Результаты моделирования

XGBoost Regressor

Выбрана за эффективность на табличных данных и скорость работы.

0.271	0.238	0.63
RMSE	MAE	R^2 Score

** Метрики ограничены скрытыми факторами (состояние кузова, ДТП), которые не указаны в параметрах. Модель показывает стабильные результаты в среднем ценовом сегменте.*



Савельева: реализация задачи train в Airflow.

Турсунов: добавление задачи evaluate и контроль успешности этапов обучения через панель мониторинга Airflow

ML-пайплайн (DVC):

preprocess

Очистка данных, кодирование, нормализация. Выход: `data/processed/dataset.csv`

split

Разделение датасета на тренировочный и тестовый (сохраняется в DVC).

train_baseline

Обучение модели XGBoost. Выход: `models/xgboost_model.pkl`.

evaluate_baseline

Оценка модели, логирование метрик и артефактов в MLflow.

Савельева: описание и реализация шагов preprocess и train.

Турсунов: настройка этапов evaluate и register.

CI/CD Pipeline (GitHub Actions)

Commit & Hooks

Testing

DVC Repro

Docker build and push

Deploy

Commit & Hooks:
Линтинг и форматирование кода

Testing:
Запуск unit-тестов через pytest

DVC Repro:
Воспроизведение пайплайна обработки данных.

Docker build and push:
Сборка Docker-образа приложения и загрузка образа в Docker Hub Registry.

Deploy:
Тестовое развертывание в кластер Kubernetes (KIND).

Жмаев: настройка базового CI (ci.yml) и расширение пайплайна для сборки и деплоя.
Савельева: настройка деплоя в Kubernetes из CI и обновление образов.
Турсунов: тестирование полного цикла от merge до обновления сервиса.

Инструменты разработки и качества

Pre-commit Hooks

Автоматический линтинг и форматирование кода:

- Black & Flake8
- Trailing whitespace
- Check YAML

Package Management

Использование uv как быстрой альтернативы pip. Ускоряет работу с зависимостями и виртуальными окружениями.

Orchestration

Airflow DAGs для регулярных задач:

- Обучение модели
- Ежедневный детектинг дрифта данных

Турсунов: написание функциональных тестов API через pytest.

Савельева: настройка conftest.py для тестирования и Docker-окружения.

Жмаев: подготовка README и настройка Airflow для ежедневных задач.

Описание API (FastAPI)

Турсунов: разработка основной логики REST API и эндпоинта /predict.

Жмаев: тестирование API с помощью curl и подготовка документации.

Основные Эндпоинты

Документация доступна через Swagger UI.

Parameters

No parameters

Request body required

Example Value | Schema

```
{
  "mark": "string",
  "model": "string",
  "super_gen_name": "string",
  "body_type_type": "string",
  "transmission": "string",
  "engine": "string",
  "year": 0,
  "mileage": 0,
  "color": "string",
  "owners": "string",
  "region": "string",
  "gear_type": "string",
  "steering_wheel": "string",
  "complectation": "string",
  "power": 0,
  "displacement": 0
}
```

Car Price Prediction API 2.0 OAS 3.1

/openapi.json

default

- GET /metrics Metrics

Car API

- GET /api/v1/params Get Params Endpoint
- GET /api/v1/marks Get Marks
- GET /api/v1/steering_wheels Get Steering Wheels
- GET /api/v1/owners Get Owners
- GET /api/v1/regions Get Regions
- GET /api/v1/gear_types Get Gear Types
- GET /api/v1/colors Get Colors
- POST /api/v1/models Get Models
- POST /api/v1/gens Get Gens
- POST /api/v1/bodies Get Bodies
- POST /api/v1/complectations Get Complectations
- POST /api/v1/transmissions Get Transmissions
- POST /api/v1/engines Get Engines
- POST /api/v1/years Get Years
- POST /api/v1/predict Predict

Мониторинг и observability

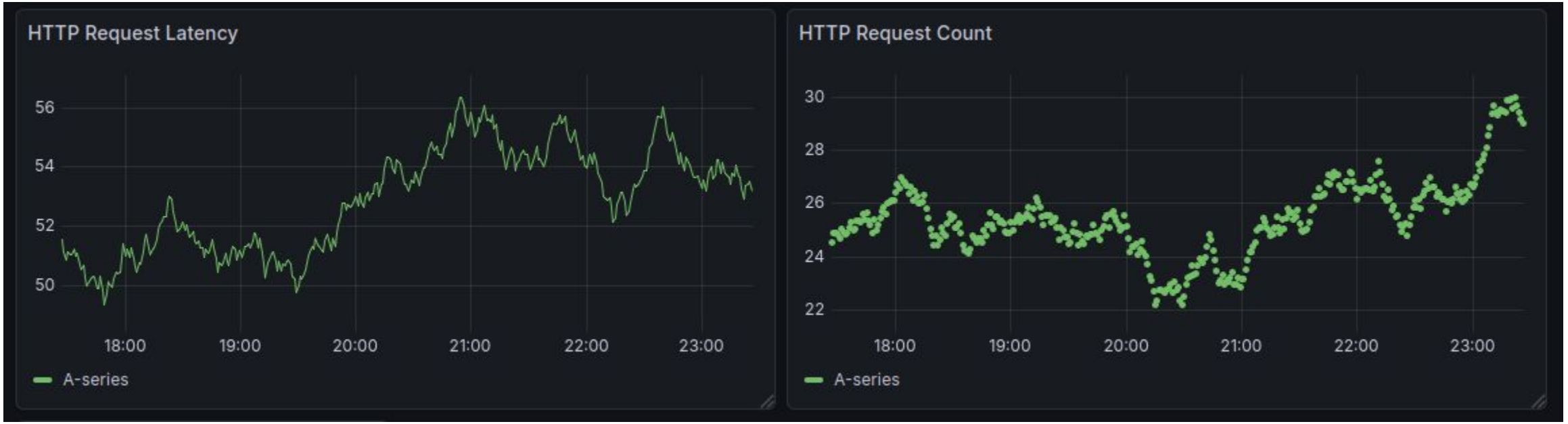
Жмаев: добавление метрик Prometheus в приложение FastAPI.
Савельева: настройка Prometheus и конфигурация сбора метрик.
Турсунов: создание дашбордов в Grafana и проверка визуализации данных.

Инструменты

- Prometheus: сбор метрик с микросервисов.
- Grafana: визуализация дашбордов.

Отслеживаемые показатели

- Скорость ответа API и количество запросов.
- Системные метрики (CPU/Memory).
- Метрики качества модели (через Airflow drift detection).



Автоматизация и оркестрация Apache Airflow

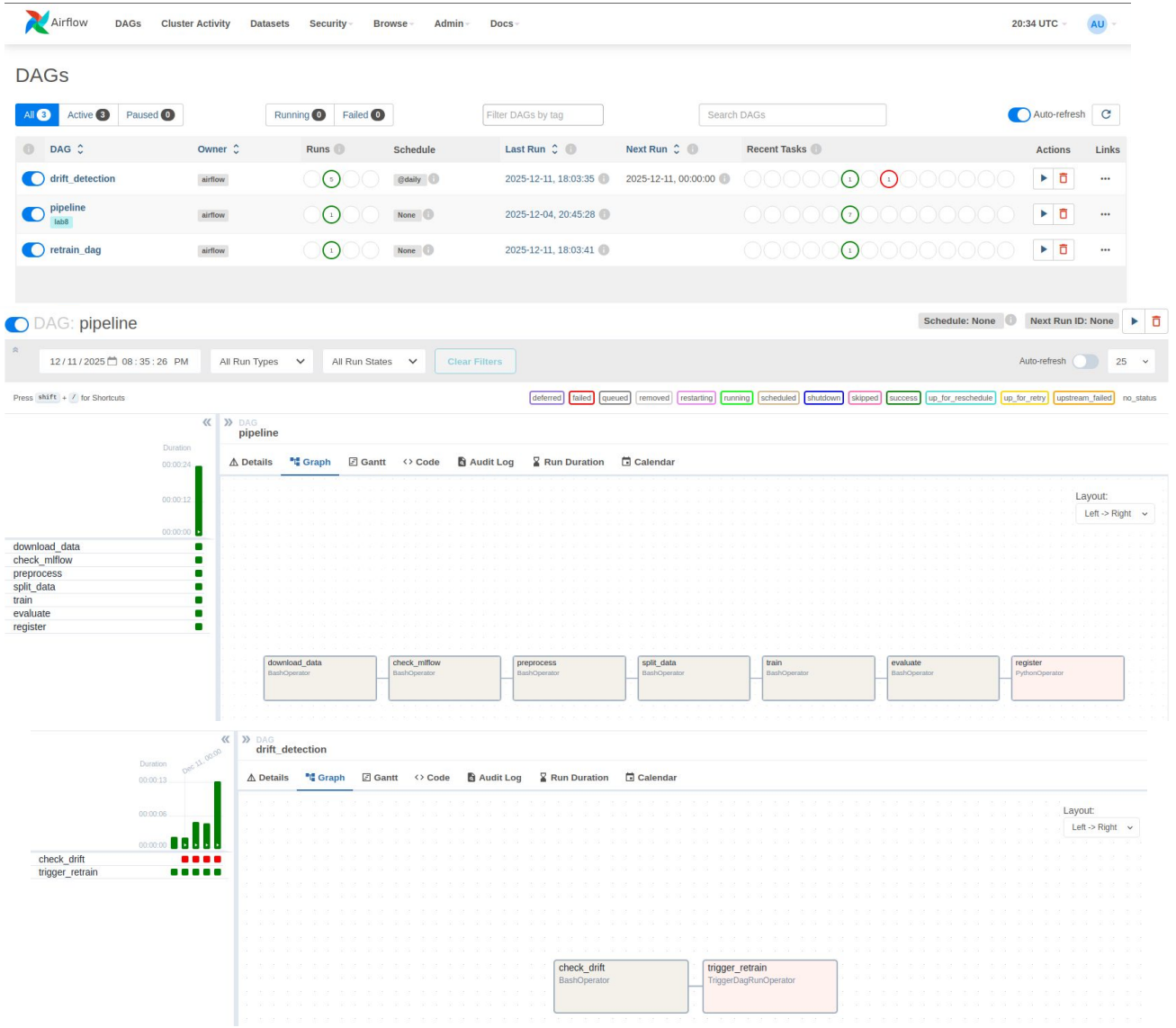
Инструменты

- Настроен DAG для ежедневного обнаружения дрифта данных (Data Drift)
- Мониторинг: сравнение распределения признаков (пробег, год) между тренировочным и текущим production-набором.
- Триггер: если дрифт превышает установленный порог, Airflow может автоматически запустить DAG переобучения.

Жмаев: реализация скрипта для детекции дрейфа признаков.

Савельева: встраивание проверок дрейфа в Airflow и настройка порогов.

Турсунов: настройка автоматического триггера переобучения при дрейфе.



Проблемы и вызовы

Данные и признаки

- Низкое качество исходных данных (объявления пользователей).
- Отсутствие информации о скрытых факторах (ДТП, состояние).
- Высокая динамика цен, требующая частого переобучения.

MLOps интеграция

- Сложность связки DVC-Feast-MLflow в едином пайплайне.
- Настройка тестового деплоя в Kubernetes KIND.
- Обеспечение низкой задержки API при инференсе из Feast.

Тестирование и контроль качества кода

API Testing

- Pytest: используется для модульного тестирования API.
- Мокирование: для быстрого тестирования логики API без обращения к внешним сервисам (Feast/MLflow).

Pre-commit Hooks

Настроены хуки для автоматического контроля качества:

- Форматирование: Black.
- Линтинг: Flake8.
- Проверка синтаксиса (YAML, конечные пробелы, большие файлы).

Турсунов: модульное тестирование API и использование моков.

Савельева: настройка хуков качества (Black, Flake8).

Дорожная карта

Улучшение модели (1-2 месяца)

- Внедрение неструктурированных данных: анализ текста объявлений (NLP) и фотографий (CV).
- Исследование альтернативных моделей (CatBoost, Нейронные сети) для повышения точности.
- Автоматизированный поиск и добавление новых, более релевантных фичей.

Развитие MLOps и инфраструктуры (3-4 месяца)

- Разработка фреймворка для A/B-тестирования новых версий модели.
- Расширение дашбордов Grafana: добавление метрик Feature Drift и Model Degradation.

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 K8s.

Команда

Турсунов Руслан: разработал REST API на FastAPI, внедрил систему функционального тестирования, настроил автоматическое переобучение моделей и создал аналитические дашборды в Grafana.

Жмаев Вячеслав: отвечал за инфраструктурный слой и автоматизацию: развертывание среды Airflow, конфигурацию Feature Store (Feast), настройку мониторинга в Prometheus и реализацию механизмов детекции дрейфа данных.

Савельева Елизавета: обеспечила контейнеризацию и оркестрацию процессов: разработала Docker-манифесты и Kubernetes-сервисы, реализовала основные задачи по препроцессингу и обучению в DAG-пайплайнах, а также настроила CD-процессы.