

Проект "Calorie Tracker"

Команда 50, тема №32 «Дневник калорий»

Описание структуры проекта

Проект организован в виде модульной архитектуры, разделённой на папки и файлы. Репозиторий содержит несколько веток, каждая из которых соответствует определённым этапам разработки и функционалу.

Ветки репозитория:

1. **main:**
 - Основная ветка, содержащая информацию о проекте.
2. **EDA_and_paper_analysis:**
 - В ветке содержится EDA по данным и анализ архив-статьи.
3. **InceptionV3_train:**
 - Обучение модели InceptionV3 (одной из моделей бейзлайна).
4. **VLM_baseline:**
 - Тестирование модели llama3.2-vision в качестве бейзлайна.
5. **YOLO_train_Streamlit_service:**
 - Обучение модели YOLO
 - Реализация Streamlit-приложения и интеграция с моделями YOLO, InceptionV3, open source VLM.
 - Содержит:
 - Streamlit-файлы в папке streamlit_app.
 - Логику API в папке service/api.
 - Подготовленные веса для YOLO-модели в service/api/models.
 - Дообучение модели YOLO в файле Modeling.ipynb
6. **Docker_build:**
 - Ветка для настройки и тестирования контейнеризации с помощью Docker.
 - Включает:
 - Dockerfile и docker-compose.yml для создания и управления контейнерами.
 - Инструкции по сборке и запуску контейнеров.
 - Дополнительные данные и зависимости, необходимые для сборки.

Детализированная структуры папок и файлов в ветке

«YOLO train Streamlit service»:

- **/service:** Основной backend-сервис.
 - **main.py:** Главный файл запуска Uvicorn, содержащий роутинг и основные функции API.
 - **/api:** Логика API.
 - **fit/:** дообучение
 - **train/:** изображения для реализации функции дообучения модели.
 - **val/:** изображения для валидации.
 - **models/:** Папка для хранения весов моделей и связанных файлов.

- **v1:** Функции, используемые в API (например, предобработка данных).
 - **data/:** Папка для хранения входных и обучающих данных.
 - **tests/:** Модульные тесты для API.
- **/streamlit_app:** Streamlit-приложение (frontend).
 - **app.py:** Основной файл приложения Streamlit, реализующий пользовательский интерфейс.
- **requirements.txt:** Список зависимостей для установки.

Детализированная структура папок и файлов в ветке «Docker build»:

- **Dockerfile.fastapi:** предназначен для сборки контейнера, содержащего backend-сервис, реализованный на FastAPI.
- **Dockerfile.streamlit:** используется для сборки контейнера с frontend-приложением, основанным на Streamlit.
- **docker-compose.yml:** Конфигурация для запуска проекта в виде контейнеров.
- **Все остальные файлы по назначению идентичны файлам в ветке «YOLO_train_Streamlit_service».**

Описание функционала API / Streamlit-приложения

API (Backend)

API реализован с использованием фреймворка FastAPI. Реализованные методы:

- 1. Загрузка модели:**
 - Endpoint: /load
 - Описание: Загружает одну из трех моделей из соответствующей директории
- 2. Классификация изображений:**
 - Endpoint: /predict
 - Описание: Принимает изображение и возвращает класс изображения (тип блюда). Выполняет либо детекцию и классификацию, либо классификацию в зависимости от выбора модели.
- 3. Дообучение модели:**
 - Endpoint: /fit
 - Описание: Реализует дообучение модели на двух изображениях.
- 4. Результат дообучения моделей:**
 - Endpoint: /fit/results
 - Описание: Предоставляет результаты обучения выбранной модели.
- 5. EDA (Exploratory Data Analysis):**
 - Endpoint: /eda
 - Описание: Генерирует сводный отчет и визуализации по загруженному датасету.
- 6. Состояние системы:**
 - Endpoint: /status
 - Описание: Проверяет работоспособность сервиса.

7. Список текущих моделей:

- Endpoint: /loaded_models
- Описание: Возвращает список загруженных моделей

8. Удаление модели:

- Endpoint: /remove/{model_id}
- Описание: Удаление модели

9. Удаление всех моделей:

- Endpoint: /remove_all
- Описание: Удаление всех моделей

Streamlit-приложение (Frontend)

Streamlit-приложение предоставляет пользовательский интерфейс для взаимодействия с системой. Основные модули:

1. Классификация:

- Поле для загрузки изображения через кнопку «Browse files».
- Вывод результата с визуализацией.

2. Анализ данных (EDA):

- Кнопка для запуска анализа на загруженном датасете «Провести EDA датасета UECFOOD256».
- Отображение сводной информации: распределения классов, основные метрики.

3. Дообучение модели YOLO:

- Кнопка «Дообучить модель YOLO на двух картинках».

4. Проверка состояния сервиса:

- Кнопка «Проверить состояние сервиса»

5. Загрузка модели:

- Возможность загрузить новую модель через выбор из выпадающего меню и кнопку «Загрузить модель».

Инструкция по использованию

Подробная инструкция по использованию с примерами кода/команд приложена в файле README.md ветки main репозитория AIYP24-Calorie-Tracker.