Классификация транспортного средства по типу кузова

Оглавление

Описание

Годовой проект команды **№47**.

Реализованный проект представляет собой веб-сервис, состоящий из FastAPI-сервера и Streamlit-приложения для взаимодействия с моделями машинного обучения для классификации изображений автомобилей.

Состав команды

- Лесниченко Макар Олегович — @makarles, makarles
- Назаров Максим Олегович — @MONazarov, MirnMax
- Нюнин Николай Андреевич — @n1kolay177, NikolayNyunin
- Сорокин Иван Михайлович — @IVANSOR0KIN, ivansor0kin

Куратор

- Козлов Кирилл — @dedpe, KozlovKY

Структура проекта

- Файл `README.md` — основной Markdown-файл с описанием проекта ***(данный файл)***.
- Файл `checkpoints.md` — описание чекпоинтов (этапов работы) по проекту.
- Папка `EDA/` — Exploratory Data Analysis.
  - Файлы `*.ipynb` — Jupyter-ноутбуки с EDA по разным датасетам.
  - Файл `dataset.md` — описание наборов данных, выбранных для решения поставленной задачи.
  - Файл `EDA.md` — выводы на основе проведённого EDA.
  - Папка `img/` — картинки для визуализации данных.
- Модуль `models/` — модели машинного обучения.
  - Модуль `baseline/` — бейзлайн-модель.
    - Файлы `*.py` — скрипты для взаимодействия с бейзлайн-моделью.
    - Файл `best_checkpoint.pt` — чекпоинт (контрольная точка) обученной модели.
- Модуль `api/` — FastAPI-сервис.
  - Модуль `v1/` — первая версия API.
    - Файл `api_route.py` — основной скрипт с описанием методов API.
  - Файл `app.py` — корневой скрипт backend-сервиса.
- Папка `frontend/` — Streamlit-приложение.
  - Файл `streamlit_app.py` — основной скрипт frontend-сервиса.
- Папка `BOT/` — Telegram бот.
  - Файл `bot.py` — основной скрипт бота (без токена).
  - Файл `server.py` — основной скрипт серверной части бота.
  - Файл `bot_simple.py` — простой скрипт бота (картинка на вход, картинка на выход).
  - Папка `available_models` — модели машинного обучения (StanfordCarsDataset, ivan).
  - Папка `demo` — демонстрация работы бота.
- Папка `logs/` — логи приложения.
- Файлы `backend.Dockerfile` и `frontend.Dockerfile` — конфигурационные файлы Docker.
- Файл `compose.yaml` — конфигурационный файл Docker Compose.
- Файлы `.gitignore`, `.gitattributes`, `pyproject.toml` и `poetry.lock` — Служебные файлы Git и Poetry.

Запуск приложения

Собрать образы и запустить Docker-контейнеры для FastAPI и Streamlit-приложений благодаря Docker Compose можно при помощи одной команды:

```
docker compose up
```

После успешного выполнения данной команды приложения будут доступны локально по следующим адресам:

- Frontend-приложение на Streamlit — http://127.0.0.1:8501 (или http://localhost:8501).
- Backend-приложение на FastAPI — http://127.0.0.1:8000 (или http://localhost:8000).

Запуск ТГ БОТА

Прежде всего, написать @IVANSOR0KIN в Telegram / тегнуть в беседе группы (токен не был загружен в репозиторий). На данный момент (02/01/25) бот развернут локально.

Адрес бота в Telegram: @VehicleClassifierBot

Использование ТГ БОТА

Демо:

https://github.com/NikolayNyunin/Vehicle-Classification/blob/develop/BOT/demo/IMG_9624.jpg

https://github.com/NikolayNyunin/Vehicle-Classification/blob/develop/BOT/demo/IMG_9625.jpg

https://github.com/NikolayNyunin/Vehicle-Classification/blob/develop/BOT/demo/IMG_9626.jpg

Установка зависимостей

В случае, если вы планируете запускать проект без контейнеризации, рекомендуется наличие менеджера зависимостей Poetry.

Если Poetry установлен, для создания виртуального окружения и установки всех необходимых проекту зависимостей можно воспользоваться командой:

```
poetry install
```

Зависимости проекта в файле `pyproject.toml` разбиты на группы, что позволяет установить не все зависимости сразу, а только необходимые для конкретной задачи.

Реализованные группы зависимостей:

- **main** — основные зависимости, нужные большинству разделов проекта.
- **backend** — зависимости FastAPI-приложения.
- **frontend** — зависимости Streamlit-приложения.
- **analytics** — зависимости Jupyter-ноутбуков с аналитикой и экспериментами.

Таким образом, например, чтобы установить только зависимости для запуска Jupyter-ноутбуков с аналитикой, можно использовать команду `poetry install --only main,analytics`, а чтобы установить все зависимости, кроме фронтендовых, можно использовать команду `poetry install --without frontend`.

Запуск FastAPI-приложения без Docker

Для запуска FastAPI-приложения без контейнеризации можно воспользоваться командой:

```
python api/app.py
```

Запуск Streamlit-приложения без Docker

Для запуска Streamlit-приложения без контейнеризации можно воспользоваться командой:

```
streamlit run frontend/streamlit_app.py
```

Описание методов API

GET "/"

Получение информации о статусе сервиса.

Пример ответа: `200 OK`

```
{"status": "OK"}
```

POST "/api/v1/fit"

Обучение и сохранение новой модели.

Пример запроса:

```
{
  "name": "TestModel",
  "description": "Test Description",
  "hyperparameters": {
    "batch_size": 16,
    "n_epochs": 5,
    "eval_every": 100
  }
}
```

Пример ответа: `201 Created`

```
{"message": "New model trained and saved with id 1"}
```

POST "api/v1/fine_tune"

Дообучение существующей модели.

Пример запроса:

```
{
  "id": 0,
  "name": "TestModel",
  "description": "Test description",
  "hyperparameters": {
    "batch_size": 16,
    "n_epochs": 1,
    "eval_every": 100
  }
}
```

Пример ответа: `201 Created`

```
{"message": "Model with id 0 fine-tuned and saved with id 1"}
```

POST "api/v1/predict"

Получение предсказаний при помощи выбранной модели.

Пример запроса: одно или несколько изображений.

Пример ответа: `200 OK`

```
[
  {
    "class_id": 6,
    "class_name": "estate",
    "confidence": 0.8
  }
]
```

GET "api/v1/models"

Получение списка сохранённых моделей.

Пример ответа: `200 OK`

```
[
  {
    "id": 0,
    "name": "TestModel",
    "description": "Test description"
  }
]
```

POST "api/v1/set"

Выбор активной модели.

Пример запроса:

```
{"id": 0}
```

Пример ответа:

```
{"message": "Model with id 0 successfully set active"}
```

Описание Streamlit-приложения

В Streamlit-приложении реализованы 4 основных страницы для взаимодействия с API.

1. Загрузка датасета и EDA

2. Обучение/Дообучение

3. Модели

4. Инференс