

Документация проекта

Структура папок проекта

Проект организован следующим образом:

MVP/

Основная папка проекта.
Содержит папки для серверной и клиентской части, а также вспомогательные данные и конфигурацию.

Вложенные папки:

fastapi/

Содержит серверную часть на FastAPI.

Файлы:

- `api_mvp.py`: реализация API.
- `ml_utils.py`: вспомогательные функции для обработки данных и обучения моделей.
- `Dockerfile`: инструкции для контейнеризации FastAPI.
- `pipeline_h.pk1` и `pipeline_r.pk1`: предобученные модели.
- `requirements.txt` - зависимости для FastAPI.
- `logs/`: папка для логов.

streamlit/

Содержит клиентскую часть на Streamlit.

Файлы:

- `streamlit_mvp.py`: основной код приложения Streamlit.
- `requirements.txt`: зависимости для Streamlit.
- `logs/`: папка для логов Streamlit.

sample_data/

Папка с тестовыми данными.

Файлы:

- `data_sample.parquet`: образец датасета для загрузки.

Корневые файлы:

- `main_mvp.py`: код для одновременного запуска FastAPI и Streamlit.
- `docker-compose.yml`: конфигурация для запуска приложения в Docker.

2. Описание функционала API

Основные эндпоинты FastAPI

1. **GET /**
 - Проверка состояния сервера.
 - **Ответ:** { "message": "Мы работаем" }
2. **POST /preprocess**
 - Выполняет предобработку загруженного датасета.
 - **Вход:** файл в формате `.parquet`.
 - **Ответ:** сообщение о статусе предобработки.
3. **POST /fit**
 - Обучает новую модель с заданной конфигурацией.
 - **Вход:** JSON с параметрами модели.
 - **Ответ:** список ID обученных моделей.
4. **POST /predict**
 - Выполняет предсказание на новых данных.
 - **Вход:** JSON с ID модели и данными для предсказания.
 - **Ответ:** JSON с предсказаниями.
5. **GET /models**
 - Возвращает список всех доступных моделей.
 - **Ответ:** JSON с массивом ID и типов моделей.
6. **POST /set**
 - Устанавливает текущую модель для предсказания.
 - **Вход:** ID модели.
 - **Ответ:** подтверждение установки модели.
7. **POST /plot_learning_curve**
 - Генерирует графики кривых обучения для моделей.
 - **Вход:**
 - `id` (строка): ID модели.
 - `cv` (число): Количество фолдов для кросс-валидации.
 - `scoring` (строка): Метрика для оценки (например, `accuracy`).
 - **Ответ:** JSON с URL графиков для модели `rating` и `hubs`.

Логирование

- Логи записываются в файл `logs/app.log` с ротацией.
- Формат логов: `[время] уровень - имя модуля - сообщение`.

3. Описание функционала Streamlit-приложения

Основные страницы

1. **Upload Dataset**
 - Загрузка датасета в формате `.parquet`.
 - Отображение первых строк и информации о данных (размерность, типы данных).
 - Отправка данных для предобработки через эндпоинт `/preprocess`.
2. **EDA (Exploratory Data Analysis)**
 - **Агрегированная информация:** основные статистики (среднее, медиана, максимум, минимум, стандартное отклонение) по числовым колонкам.
 - **Распределение данных:** распределение рейтингов статей.
 - **Топ-10 частотных слов:** анализ наиболее популярных токенов в текстах.
 - **Облако слов:** генерация облаков слов для текстов или тегов.
 - **Распределение частей речи:** графики частот частей речи для токенов.
3. **Train New Model**
 - Настройка гиперпараметров:
 - Для `TfidfVectorizer`: `max_features`, `max_df`, `min_df`.

- Параметры TfIdfVectorizer:
- max_features** :
 - Ограничивает максимальное количество признаков (токенов), которые будут использоваться в модели.
 - Признаки отбираются по их частотности. Например, если значение `max_features=1000`, то будут выбраны 1000 наиболее часто встречающихся слов.
 - min_df** :
 - Минимальная доля документов, в которых должен встречаться токен, чтобы быть включенным в модель.
 - Значение задается как доля (например, `0.01`).
 - Пример: при `min_df=0.01` токены, встречающиеся менее чем в 1% документов, будут игнорироваться.
 - max_df** :
 - Максимальная доля документов, в которых может встречаться токен.
 - Значение задается как доля (например, `0.90`).
 - Пример: при `max_df=0.90` токены, встречающиеся более чем в 90% документов, будут считаться "мусорными" и игнорироваться.
 - Для классификатора (`LogisticRegression` или `SVC`): параметры регуляризации, итераций и решателя.
 - Задание уникального ID для модели.
 - Отправка конфигурации на сервер для обучения новой модели через `/fit`.
 - Model Info**
 - Просмотр списка всех доступных моделей, обученных ранее.
 - Установка выбранной модели текущей через эндпоинт `/set`.
 - Генерация и просмотр кривых обучения для выбранной модели через эндпоинт `/plot_learning_curve`.
 - Inference**
 - Выполнение предсказаний на новых данных с использованием выбранной модели.
 - Отправка данных для предсказания через эндпоинт `/predict`.
 - Отображение результатов предсказаний в виде таблицы.

4. Структура тестового датасета `data_sample.parquet`, на котором тестируем сервис

Описание колонок

- author** (строка):
 - Имя автора статьи.
- publication_date** (дата/время):
 - Дата и время публикации статьи.
- hubs** (строка):
 - Тематические хабы (категории), к которым относится статья.
- comments** (числовое значение):
 - Количество комментариев к статье.
- views** (числовое значение):
 - Количество просмотров статьи.
- url** (строка):
 - Ссылка на статью.
- reading_time** (числовое значение):
 - Время, необходимое для чтения статьи, в минутах.
- indiv/comp** (строка):
 - Тип автора (индивидуальный или корпоративный).
- bookmarks_cnt** (числовое значение):
 - Количество добавлений статьи в закладки.
- text_length** (числовое значение):
 - Длина текста статьи в символах.
- tags_tokens** (список строк):
 - Токенизированные теги статьи.
- title_tokens** (список строк):
 - Токенизированное название статьи.
- rating_new** (числовое значение):
 - Рейтинг статьи.
- text_tokens** (список строк):
 - Токенизированное содержание текста статьи.
- text_pos_tags** (список строк):
 - Части речи для токенов текста (например, `NOUN`, `VERB`, `ADJ`).
- rating_level** (строка):
 - Уровень рейтинга статьи (например, `very positive`, `neutral`).

Пример строки

author	publication_date	hubs	comments	views	url	reading_time	indiv/company	bookmarks_cnt	text_length	tags_tokens	title_tokens	rating_new	te
krig	2010-01-21 13:11:17+00:00	Мессенджеры	49	1000	https://habr.com/ru/articles/81478/	1.0	individual	7.0	374	['skype', 'linux']	['skype', 'beta', 'linux']	44.0	['с

5. Инструкция по использованию

Установка и запуск (через Докер)

- Запустите Докер, перейдите в папку `MVP` и выполните:

```
docker-compose build
```

- После сборки образов выполните:

```
docker-compose up
```

- Приложение запустится по адресу:

<http://localhost:8501/>

Установка и запуск (через скрипт)

1. Установите все зависимости:

◦ Для FastAPI: перейдите в папку `fastapi` и выполните:

```
pip install -r requirements.txt
```

◦ Для Streamlit: перейдите в папку `streamlit` и выполните:

```
pip install -r requirements.txt
```

2. Настройка приложения для локального запуска:

◦ Откройте файл `streamlit/streamlit_mvp.py`.

◦ **Закомментируйте** строку с указанием URL для запуска в Docker:

```
API_URL = "http://fastapi:8080"
```

◦ **Раскомментируйте** строку с локальным API:

```
API_URL = "http://127.0.0.1:8080"
```

3. Запустите приложение:

◦ Перейдите в корневую папку проекта `MVP/`.

◦ Выполните команду:

```
python main_mvp.py
```

Демонстрация работы приложения

Для просмотра демонстрации работы приложения перейдите по ссылке:
[Демонстрация приложения](#)