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.pkl` и `pipeline_r.pkl`: предобученные модели.
- `reqirements.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:

1. `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.

2. `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.

3. `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`.

4. **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`.

5. **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

1. `author` (строка):
   - Имя автора статьи.
2. `publication_date` (дата/время):
   - Дата и время публикации статьи.
3. `hubs` (строка):
   - Тематические хабы (категории), к которым относится статья.
4. `comments` (числовое значение):
   - Количество комментариев к статье.
5. `views` (числовое значение):
   - Количество просмотров статьи.
6. `url` (строка):
   - Ссылка на статью.
7. `reading_time` (числовое значение):
   - Время, необходимое для чтения статьи, в минутах.
8. `individ/company` (строка):
   - Тип автора (индивидуальный или корпоративный).
9. `bookmarks_cnt` (числовое значение):
   - Количество добавлений статьи в закладки.
10. `text_length` (числовое значение):
    - Длина текста статьи в символах.
11. `tags_tokens` (список строк):
    - Токенизированные теги статьи.
12. `title_tokens` (список строк):
    - Токенизированное название статьи.
13. `rating_new` (числовое значение):
    - Рейтинг статьи.
14. `text_tokens` (список строк):
    - Токенизированное содержание текста статьи.
15. `text_pos_tags` (список строк):
    - Части речи для токенов текста (например, `NOUN`, `VERB`, `ADJ`).
16. `rating_level` (строка):
    - Уровень рейтинга статьи (например, `very positive`, `neutral`).

Пример строки

author	publication_date	hubs	comments	views	url	reading_time	individ/company	bookmarks_cnt
krig	2010-01-21 13:11:17+00:00	Мессенджеры	49	1000	https://habr.com/ru/articles/81478/	1.0	individual	7.0

---

5. Инструкция по использованию

Установка и запуск

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** строчку с указанием URL для запуска в Docker:

     ```
     API_URL = "http://fastapi:8000"
     ```

   - **Раскомментируйте** строчку с локальным API:

     ```
     API_URL = "http://127.0.0.1:8000"
     ```

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