Документация модулей и файлов

Модуль `app.py` — FastAPI-приложение

Назначение

Файл `app.py` содержит веб-сервис на базе FastAPI, который:

- принимает CSV-файл с данными пациентов;
- выполняет предобработку и предсказание обученной моделью;
- возвращает результат в формате JSON.

Модуль выступает в роли API-слоя между пользователем и моделью.

Эндпоинты

GET `/health` Проверка работоспособности сервиса.

Ответ:

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

POST `/predict` Принимает CSV-файл, выполняет предсказание модели и возвращает результат.

Параметры запроса:

- `file`: `UploadFile` — CSV-файл с данными пациентов.

Структура ответа: список объектов вида:

```
[
  {"id": 1, "prediction": 0},
  {"id": 2, "prediction": 1}
]
```

Последовательность обработки запроса

1. Чтение CSV-файла в `pandas.DataFrame`.
2. Логирование размера входных данных.
3. Вызов метода `model.pred_target()` для предобработки и инференса.
4. Формирование итогового `DataFrame` с предсказаниями.
5. Возврат JSON-ответа клиенту.

Модуль `logging_file.py` — логирование

Функция `setup_logging()`

Функция `setup_logging()` настраивает глобальный формат логов и вывод сообщений в консоль.

Формат логов:

`2025-11-16 20:25:11 | preprocessor | INFO | Сообщение`

Функция вызывается автоматически при старте приложения и обеспечивает единообразное логирование во всех модулях.

Модуль `model.py` — класс `Model`

Назначение

Класс `Model` отвечает за:

- загрузку обученной модели/пайплайна из файла;
- подготовку входных данных;
- получение предсказаний по подготовленным данным.

FastAPI вызывает методы этого класса при обработке запросов на предсказание.

Инициализация

```
Model(model_pipeline_path: str)
```

Параметры конструктора:

- `model_pipeline_path` — путь к сериализованному пайплайну (например, `model_pipeline.pkl`).

Основные атрибуты:

- `self.model` — загруженная ML-модель или пайплайн (через `joblib`);
- `self.preprocessor` — экземпляр класса `Preprocessor`.

Метод `pred_target`

```
pred_target(self, data: pd.DataFrame) -> np.ndarray
```

Метод выполняет полный цикл обработки данных:

1. Предобработка входных данных через `Preprocessor.transform()`.
2. Логирование размера и списка колонок после предобработки.
3. Получение предсказаний с помощью `self.model.predict()`.
4. Возврат массива предсказаний.

Параметры:

- `data` — исходные данные пользователя в формате `pandas.DataFrame`.

Возвращаемое значение:

- `y_pred` — массив предсказаний (`numpy.ndarray`, формат зависит от модели).

Модуль `preprocessor.py` — класс `Preprocessor`

Назначение

Класс `Preprocessor` отвечает за преобразование входного `DataFrame` к виду, который ожидает обученная модель. В модуле реализована ручная предобработка признаков.

Метод `transform`

```
transform(self, data: pd.DataFrame) -> pd.DataFrame
```

Основной метод предобработки данных.

Шаги обработки:

1. `fillna(0)` — заполнение пропусков нулевыми значениями.
2. Преобразование пола (`Sex`) в бинарные значения (`Male` $\rightarrow$ 1.0, `Female` $\rightarrow$ 0.0).
3. Преобразование ряда медицинских признаков к строковому типу, если это требуется логикой модели.
4. Удаление признаков с утечкой целевого признака (например, `CK-MB`, `Troponin`).
5. Удаление ненужных служебных столбцов (`Unnamed: 0`, `id` и т. п.).
6. Логирование успешного завершения трансформации.

Параметры:

- `data` — сырые пользовательские данные (`pandas.DataFrame`).

Возвращаемое значение:

- очищенный и преобразованный `DataFrame`, готовый для передачи в модель.