Документация по модулям и файлам проекта

Сервис предсказания риска сердечного приступа Дата: 18 декабря 2025 г.

Обзор проекта

Проект реализует полный цикл разработки ML-модели для предсказания риска сердечного приступа на основе данных пациентов. Использованы лучшие практики: **ООР** для структурирования кода, воспроизводимость через фиксированный pipeline, интеграция в веб-сервис на FastAPI.

Основные компоненты:

- Обработка данных (Preprocessor)
- Модель (Model с joblib pipeline)
- API и веб-интерфейс (app.py)
- Логирование (logging_file.py)
- Тестирование (test.py)

1. Общая структура проекта

```
project_root/
├── data/
│   ├── heart_train.csv            # Обучающие данные
│   └── heart_test.csv             # Тестовые данные
├── models/
│   └── model_pipeline.pkl         # Сохранённый полный scikit-learn Pipeline
├── static/
│   └── style.css                  # Стили для веб-интерфейса
├── templates/
│   └── index.html                 # Шаблон главной страницы (Jinja2)
├── logs/
│   └── heart_risk_service.log     # Логи приложения (создаётся автоматически)
├── tests/
│   ├── test.py                   # Скрипт автоматической проверки качества предсказаний
│   └── heart_test_predict.csv    # Пример файла с предсказаниями
├── preprocessor.py               # Класс предобработки сырых данных
├── logging_file.py              # Настройка централизованного логирования
├── model.py                     # Обёртка над сохранённой моделью
├── app.py                       # Основное FastAPI-приложение (API + веб-интерфейс)
└── requirements.txt             # Зависимости
```

Все ключевые компоненты вынесены в отдельные модули.

2. Описание модулей и файлов

preprocessor.py

Назначение: Внешняя предобработка сырых данных перед подачей в pipeline. **Структура**:

- Класс Preprocessor (статические методы для удобства использования без экземпляров).
- Константа COLS_FOR_DROP — список колонок, которые всегда удаляются как технические или потенциально утекающие.

Основной метод:

Python

@classmethod

def transform_raw_data(cls, data: pd.DataFrame) -> Tuple[Optional[pd.DataFrame], int]:

- Удаляет указанные колонки ('Unnamed: 0', 'CK-MB', 'Troponin', 'id').
- Выполняет dropna() — удаление всех строк с любыми пропусками.
- Возвращает:
    - Обработанный DataFrame (готовый для pipeline.predict()).
    - Количество удалённых строк (для мониторинга и логирования).

Почему именно так:

- CK-MB и Troponin — маркеры повреждения миокарда, измеряемые **после** события. Их наличие создаёт **утечку данных** и завышает метрики. Удаление обязательно.
- Простое удаление строк с пропусками приемлемо так как строки с пропусками в Gender испорчены: вместо male/ female стоят числовые значения [0,1] при однообразных данных в других колонках. Эти строки следует исключить.
  Так данные обрабатываются одинаково и при обучении модели, и при её использовании

logging_file.py

Назначение: Централизованная настройка логирования для всего проекта. **Структура**:

- Создаёт директорию logs/ при необходимости.
- Конфигурация через dictConfig: вывод в консоль (INFO) и файл (WARNING)
- Функции setup_logging() и get_logger(name).

Применение: Вызывается один раз в app.py при старте приложения.

model.py

Назначение: Обёртка над сохранённым полным pipeline для инференса. **Структура**:

- Класс Model.
- __init__ загружает pipeline через joblib.load().
- Метод predict_with_info(data: pd.DataFrame):
    - Применяет Preprocessor.transform_raw_data().
    - Делает предсказания через загруженный pipeline.
    - Возвращает кортеж: предсказания (np.ndarray int), обработанные данные, количество удалённых строк.

app.py

Назначение: Основное FastAPI-приложение — предоставляет веб-интерфейс и API для получения предсказаний. **Структура**:

- Настройка FastAPI, монтирование статики и шаблонов.
- Вызов setup_logging().
- Загрузка модели через класс Model (путь из переменной окружения или по умолчанию).
- Эндпоинты:
    - GET / и GET /health — главная страница и проверка статуса.
    - POST /predict — веб-форма: возвращает HTML с таблицей результатов.
    - POST /api/predict — API: streaming JSON с предсказаниями.
- Валидация входного CSV, сохранение соответствия id → prediction после предобработки.

index.html и style.css

Назначение: Простой и понятный веб-интерфейс.

- Форма загрузки CSV.
- Таблица результатов с цветовой индикацией риска (красный — высокий, зелёный — низкий).
- Информация об удалённых строках.
- Кнопка сохранения результатов

test.py

Назначение: Автоматическая проверка качества предсказаний на скрытом тестовом наборе. **Структура**:

- Читает два файла: предсказания студента и эталонные ответы.
- Проверяет формат и длину.
- Выводит отчет (precision, recall, **F1-score**, accuracy).

Метрика: **F1-score** — основная для оценки из-за дисбаланса классов и медицинской значимости минимизации ложных отрицаний.

heart_test_predict.csv

Пример корректного файла: две колонки id,prediction,.

3. Способ применения и запуска проекта

1. **Установка зависимостей** (рекомендуется виртуальное окружение):

Bash

pip install -r requirements.txt

2. **Размещение модели**:
    - Поместить model_pipeline.pkl в директорию models/.

- Или задать переменную окружения MODEL_PATH.

3. **Запуск сервиса**:

Bash

python app.py --port 8999 --host 0.0.0.0

По умолчанию: http://localhost:8999

4. **Использование**:
    - Веб-интерфейс: открыть в браузере, загрузить CSV с колонкой id.
    - API: POST-запрос на /api/predict с файлом.