

Документация проекта

Общая информация

Сервис хранится в папке Services и разделен на два основных компонента: бэкэнд (папка fastapi_app) и фронтэнд (папка streamlit_app). В папке Services также находится файл docker-compose.yml — файл конфигурации для запуска контейнеров в Docker Compose.

Бэкэнд реализует API для обработки данных о вакансиях и предсказания заработной платы.

Фронтэнд предоставляет пользовательский интерфейс для взаимодействия с API и визуализации данных.

Требования к окружению и запуск проекта:

- Версия Python: для работы проекта необходим Python версии 3.9 или выше.
- Необходимые библиотеки для установки:

Для работы API и Streamlit-части

- uvicorn (для запуска приложения на FastAPI)
- fastapi (веб-фреймворк для создания API на Python)
- pydantic (для валидации данных и управления схемами данных с использованием аннотаций типов Python)
- typing (для аннотаций типов)
- joblib (для эффективного сохранения и загрузки объектов Python)
- scikit-learn==1.6.0 (для машинного обучения)
- pandas (для анализа и обработки данных)

- numpy (для научных вычислений)
- matplotlib (для визуализации данных)
- wordcloud (для создания облаков слов - визуализаций)
- streamlit (для создания веб-приложений для визуализации данных и МО)
- requests (для работы с HTTP-запросами)
- plotly (для создания интерактивных графиков)
- seaborn (для визуализации данных на основе matplotlib)

- Локальный запуск:

pip install -r requirements.txt (установка зависимостей)

uvicorn fastapi_app.main:app --host 0.0.0.0 --port 8000 --reload (запуск бэкэнда)

streamlit run streamlit_app/streamline_service.py (запуск фронтэнда)

- Запуск через Docker:

Убедитесь, что установлен Docker & Docker Compose

В корневой папке проекта выполните команду: docker-compose up --build

I. API

1. Описание структуры проекта

Проект состоит из нескольких файлов, каждый из которых выполняет свою уникальную функцию:

- `main.py`: Основной файл приложения, реализующий API на базе FastAPI. Он содержит маршруты для обучения модели и предсказания заработной платы на основе данных о вакансиях.
- `preprocessing.py`: Файл, содержащий функции для предобработки данных. Он включает в себя функции для преобразования названий вакансий в категории и подготовки данных для обучения модели.
- `schemas.py`: Файл, описывающий структуры данных, используемые в API. Он определяет модели для входных и выходных данных, включая вакансии, запросы на обучение модели и предсказания.
- `jobmarket_model.pkl`: Сохраненная модель, используемая для предсказания заработной платы. Этот файл необходим для работы эндпоинта `/predict`, так как он содержит обученные параметры модели.
- `requirements.txt`: Список необходимых библиотек для установки.
- `Dockerfile`: Файл для создания Docker-образа приложения.

2. Описание функционала API

1. */fit*

Обучает модель на основе предоставленных данных о вакансиях.

- `FitRequest`: включает в себя данные о вакансиях и конфигурацию модели.
- `FitResponse`: сообщение о статусе обучения модели.

2. */predict*

Выполняет предсказание заработной платы на основе данных о вакансиях с использованием обученной модели.

- `PredictRequest`: включает в себя идентификатор модели и данные о вакансиях.
- `PredictionResponse`: список предсказанных значений заработной платы.

3. */model_info*

Возвращает информацию о текущей модели, включая коэффициенты, статистические показатели обучающей и тестовой выборок.

- `ModelInfoResponse`: содержит информацию о модели, такую как коэффициенты и статистические метрики.

4. */models*

Возвращает список всех доступных моделей.

- `ModelsResponse`: содержит массив объектов, представляющих доступные модели и их идентификаторы.

5./set

Устанавливает активную модель для использования в предсказаниях.

- `SetModelRequest`: включает идентификатор модели, которую необходимо установить.
- `SetModelResponse`: сообщение о статусе установки активной модели.

6. Логирование через logging:

В приложении настроено логирование, которое позволяет отслеживать запущенные команды и ошибки. Логи сохраняются в папке `logs` (2025-01-03 05:13:09,674 - JobMarketAPI

- INFO - Loading default model...).

3. Инструкция по использованию API

1. Запуск приложения:

- Убедитесь, что у вас установлен Python и необходимые библиотеки.

2. Обучение модели:

- Отправьте POST-запрос на endpoint `/fit` с данными о вакансиях и конфигурацией модели в формате JSON.

3. Предсказание заработной платы:

- Отправьте POST-запрос на endpoint `/predict` с идентификатором модели и данными о вакансиях в формате JSON.
- Получите предсказанные значения заработной платы.

II. Streamlit

1. Описание структуры проекта

Проект включает в себя Streamlit-приложение, которое взаимодействует с API для анализа вакансий и предсказания заработной платы. Основной файл приложения отвечает за пользовательский интерфейс, обработку взаимодействий и визуализацию данных.

Файлы проекта:

- `requirements.txt`: Список необходимых библиотек для установки.
- `config.toml`: Конфигурационный файл для Streamlit.
- `streamline_service.py`: Главный файл Streamlit-приложения.
- `Dockerfile`: Файл для создания Docker-образа приложения.

2. Описание функционала Streamlit-приложения

1. Загрузка данных:

- Пользователь может загрузить CSV-файл с данными о вакансиях.

2. Анализ данных:

- Приложение предоставляет возможность проводить исследовательский анализ данных, включая:

- Вывод информации о загруженных данных.
- Анализ пропусков и дубликатов и визуализация распределения зарплат.
- Анализ профессиональных ролей и навыков.

3. Обучение модели:

- Пользователь может настроить гиперпараметры для обучения модели и отправить запрос на обучение модели через API.

- Возможность отображения кривых обучения.

4. Предсказание заработной платы:

- Пользователь может загрузить новый CSV-файл для предсказания заработной платы с использованием обученной модели.

3. Инструкция по использованию

1. Установка зависимостей:

- Убедитесь, что у вас установлен Python и необходимые библиотеки.

2. Запуск приложения:

- Запустите приложение

3. Использование приложения:

- Загрузите CSV-файл с данными о вакансиях.
- Выберите нужный раздел в меню (EDA, Обучение модели, Предсказание).
- Вводите параметры и нажимайте соответствующие кнопки для выполнения действий.

Для дальнейшего улучшения документации будут рассмотрены добавления следующих элементов:

- Примеры запросов: Включение примеров JSON-запросов и ответов для API.
- Обработка ошибок: Описание возможных ошибок и их решений для API и Streamlit-приложения.