

# Общая концепция

Глобально логика приложения разделена на 2 составляющие: backend и frontend. В infra-dev/ и infra/ располагаются файлы для локального развертывания при разработке и развертывания на сервере.

SERVICE/

└─ backend

└─ frontend

└─ infra-dev

└─ infra

Для запуска контейнеров локально нужно в директории SERVICE/infra-dev/ создать файл .env и поместить туда переменные из .env.dev.template, затем выполнить:

```
docker compose up -d
```

## Backend (FastApi)

### Общее

Серверная часть реализует основную логику приложения и предоставляет эндпоинты для фронтенда.

### Техническая часть

#### Основные технические особенности:

- Контейнеризация
- Логирование
- Асинхронность
- Мультипроцессинг

## Структура:

SERVICE/backend/

- |— Dockerfile
- |— api
  - | |— \_\_init\_\_.py
  - | |— utils.py
  - | |— v1
    - | | |— \_\_init\_\_.py
    - | | |— background\_tasks.py
    - | | |— trainer.py
- |— dependency.py
- |— exceptions.py
- |— main.py
- |— models
  - | |— default
    - | | |— model\_lr\_e.cloudpickle
    - | | |— model\_mnb\_e.cloudpickle
    - | | |— model\_svc\_e.cloudpickle
- |— requirements.txt
- |— serializers
  - | |— \_\_init\_\_.py
  - | |— background\_tasks.py
  - | |— trainer.py
  - | |— utils
    - | | |— \_\_init\_\_.py
    - | | |— trainer.py
- |— services

```
|   ├── __init__.py
|   ├── background_tasks.py
|   ├── trainer.py
|   └── utils
|
|   ├── __init__.py
|   └── trainer.py
|
|── settings
|
|   ├── __init__.py
|   └── app_config.py
|
|── store.py
```

#### Описание основных компонент:

1. `SERVICE/backend/main.py`

Entrypoint приложения, отвечает за запуск FastAPI сервера и осуществляет подготовку перед этим (загружает модели по умолчанию в память и создает общий пул процессов).

2. `SERVICE/backend/services`

Здесь реализован сервисный слой:

- `trainer.py` : логика, связанная с моделями
- `background\_tasks.py` : логика, связанная с хранимой информацией о фоновых задачах

В SERVICE/backend/services/utils находятся вспомогательные функции

3. `SERVICE/backend/serializers`

Тут реализованы pydantic схемы для сериализации/десериализации данных. Разделение по файлам аналогично сервисному слою.

4. `SERVICE/backend/api`

Содержит реализацию эндпоинтов

5. `SERVICE/backend/Dockerfile`

Описание Docker-образа для контейнеризации приложения

6. `SERVICE/backend/models`

Директория, в которой хранятся файлы всех обученных моделей

7. `SERVICE/backend/settings`

Здесь находятся настройки приложения

8. `SERVICE/backend/store.py`

Данный файл представляет из себя импровизированную БД и содержит словари для хранения информации о фоновых задачах, объекты моделей, загруженных в память, и информацию о всех обученных моделях

9. `SERVICE/backend/dependency.py`

Здесь находятся зависимости, внедряемые в эндпоинты

10. `SERVICE/backend/exceptions.py`

Тут добавлены кастомные ошибки

11. `SERVICE/frontend/requirements.txt`

Необходимые зависимости

### Эндпоинты:

1. api/v1/models (trainer.py)

- / (получение списка моделей и информации о них: обучены ли, выгружены ли, набор гиперпараметров)
- /fit (запуск фоновой задачи обучения модели в фоне в отдельном процессе)
- /load (добавление в пространство инференса указанной модели)
- /unload (выгрузка указанной модели из инференса)
- /predict (получение предсказаний при помощи указанной модели)
- /predict\_scores (predict\_proba / decision\_function для постройки ROC-кривых)
- /remove (удаление\* указанной модели из хранилища)
- /remove\_all (удаление\* всех моделей из хранилища)

\* - кроме моделей по умолчанию

2. api/v1/tasks (background\_tasks.py)

- / (получение списка задач с актуальными на момент запроса статусами)

# Frontend (Streamlit)

## Общее

Клиентская часть представляет собой мультистраничное Streamlit-приложение, реализующее следующий функционал в контексте задачи классификации токсичных комментариев:

- Разведочный анализ (визуализация признаков от текстовых данных в зависимости от метки класса: длина текста, части речи, частотность токенов/N-грамм, облака слов)
- Управление моделями машинного обучения (обучение, инференс, отслеживание задач на обучение, получение информации о доступных моделях)
- Сравнение моделей между собой на основе метрик качества
- Получение предсказаний

## Техническая часть

### Основные технические особенности:

- Контейнеризация
- Логирование
- Асинхронность
- Модульность
- Кэширование и session\_state

### Структура:

SERVICE/frontend/

```
|  └─ app_pages
|  |  └─ bg_tasks.py
|  |  └─ eda.py
|  |  └─ file_upload.py
|  |  └─ manage_models.py
|  |  └─ predict.py
|  |  └─ predict_scores.py
|  |  └─ train_models.py
|  └─ utils
|  └─ __init__.py
```

```
|   |— client.py
|   |— data_processing.py
|   |— streamlit_helpers.py
|— Dockerfile
|— requirements.txt
|— app.py
|— logger_config.py
```

#### **Описание основных компонент:**

##### 12. `SERVICE/frontend/app.py`

Entrypoint приложения, отвечающий за объединение страниц и маршрутизацию

##### 13. `SERVICE/frontend/app\_pages`

Директория, содержащая модули, реализующие функционал различных страниц приложения:

- `file\_upload.py`: точка входа в приложение с точки зрения пользователя, интерфейс для загрузки данных для последующего обучения или аналитики
- `eda.py`: разведочный анализ данных, визуализации
- `manage\_models.py`: управление хранящимися на сервере моделями машинного обучения
- `train\_models.py`: интерфейс для обучения новых моделей с выбором составляющих пайплайна и гиперпараметров
- `bg\_tasks.py`: просмотр состояния фоновых задач обучения моделей
- `predict.py`: выполнение предсказаний при помощи выбранной модели
- `predict\_scores.py`: сравнение моделей на основе интегральных метрик качества

##### 14. `SERVICE/frontend/utils`

Вспомогательные модули

- `client.py`: управление клиентскими запросами (взаимодействие с API).
- `data\_processing.py`: утилиты для различных проверок и преобразований данных
- `streamlit\_helpers.py`: функции для упрощения логики создания streamlit-объектов на основных страницах

##### 15. `SERVICE/frontend/logger\_config.py`

Конфигурация и инстанцирование логгера

#### 16. `SERVICE/frontend/Dockerfile`

Описание Docker-образа для контейнеризации приложения

#### 17. `SERVICE/frontend/requirements.txt`

Необходимые зависимости

## Логирование

При запуске контейнеров в директориях app/ создаются директории logs/backend и logs/frontend, в которых создаются файлы для сохранения логов. Эта директория проброшена в docker volume (как и директория с файлами моделей), так что логи сохраняются даже после остановки контейнеров.

## Инфраструктура

Фронтенд и бэкенд работают в разных контейнерах, которые запускаются с помощью docker-compose.yml. В infra-dev/ находится файл для локального развертывания, в infra/ - для развертывания на сервере. Файл для прода отличается тем, что там образы подтягиваются с DockerHub, также там закрыты от внешнего мира порты бэкенда и фронтенда. Фронтенд и бэкенд скрыты за прокси NGINX, который перенаправляет запросы в нужный сервис. Также в дальнейшем планируется настройка ci-cd для автоматического обновления кода на сервере и добавление полноценной базы данных.

Приложение было развернуто на арендованном сервере, посмотреть его можно, перейдя на <http://89.110.90.133/>. На данном этапе деплой производился руками, были добавлены на сервер файлы infra/docker-compose.yml, env-файл с переменными окружения и файл nginx.conf для конфигурации NGINX. Образы тянутся с DockerHub командой docker compose pull.

Взаимодействие с машиной происходило через SSH-соединение.

## User Guide

Описание работы интерфейса(frontend)

Главной точкой входа в интерфейс является Вкладка **Homepage/data upload** вы загружаете файл csv с токсичными комментариями - обязательно наличие столбцов toxic и comment\_text - после загрузки вы можете просмотреть dataframe загруженного файла(Мы просим по возможности использовать файл **demo data sample.csv** который лежит в корне папки SERVICE - для демонстрации работы проекта

## App for analyzing data, managing models and obtaining predictions for the task of classifying toxic comments

Upload data to analyze/train model with



Drag and drop file here

Limit 10GB per file • CSV

Browse files



demo\_data\_2.csv 38.9MB

×

## Data preview

|  |       |
|--|-------|
| comment_text   | toxic |
| She probably went back to her pedophile protector site. Apologist Dave Pierre has a sick and twisted site "the media report." Apologists there have a goal to defend the indefensible and call themselves traditional Catholics. I am angered and sickened by the lies and disrespect the media report has shown to my personal hero, Tom Doyle. If I met Pierre on the street, I am afraid that I could not control my emotions, but would gladly spend a few days in the Graybar hotel to give him the what for about his lies and remarks about Tom Doyle. I call this kind of act, traditional Irish Catholic problem resolution. (even though I am Polish) Pierre represents Tom Doyle as a villain and a traitor to the church. He calls Tom a liar and a fraud. Pandora fits right in | 1     |



Далее во вкладке **Train models** вы можете обучить одну из доступных типов моделей с гиперпараметрами которые подходят для выбранной модели(Logistic Regression, LinearSvc, Naive Bayes)

Home

Homepage/data upload

Models

Manage models

**Train models**

Predict

Compare models by ROC

Analytics

EDA

Tools

Monitor tasks

Deploy

### Model training

Select model and it's params

logistic\_regression

penalty

l2

solver

lbfgs

max\_iter

100

-

+

c

1,00

-

+

Select vectorizer and it's params

bag\_of\_words

max\_features

1

100000

min\_df

1

100

max\_df

1,00

-

+

Enter the name of your model – model\_id

Apply params and fit

После запуска обучения вы можете мониторить текущие задачи обучения во вкладке **Monitor tasks**

<

Deploy

:

Home

Homepage/data upload

Models

Manage models

Train models

Predict

Compare models by ROC

Analytics

EDA

Tools

Monitor tasks

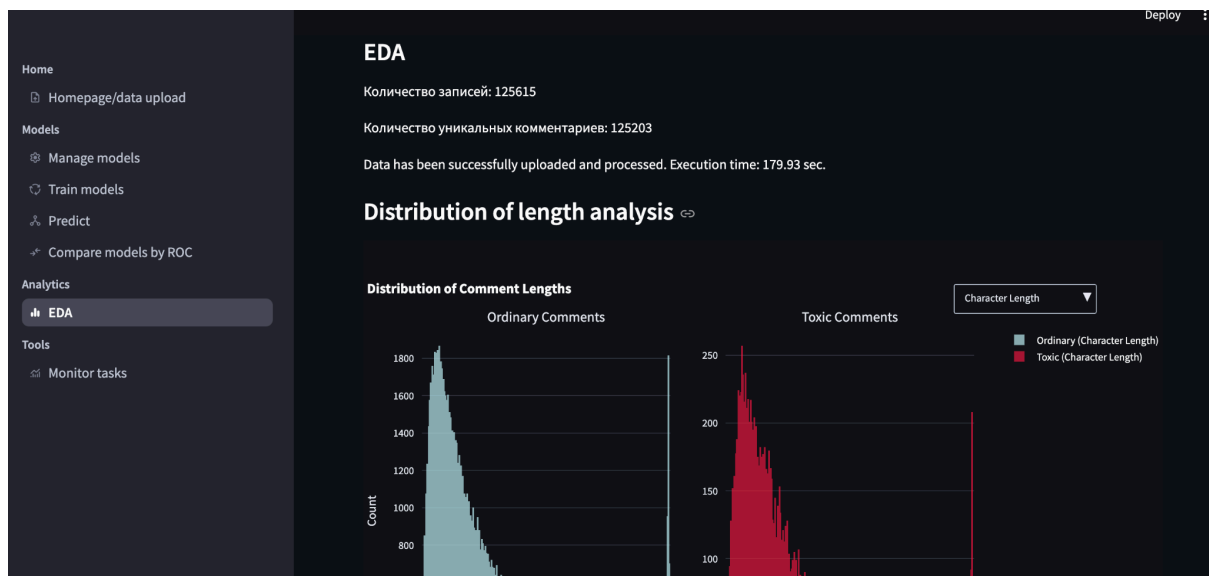
Background tasks monitor

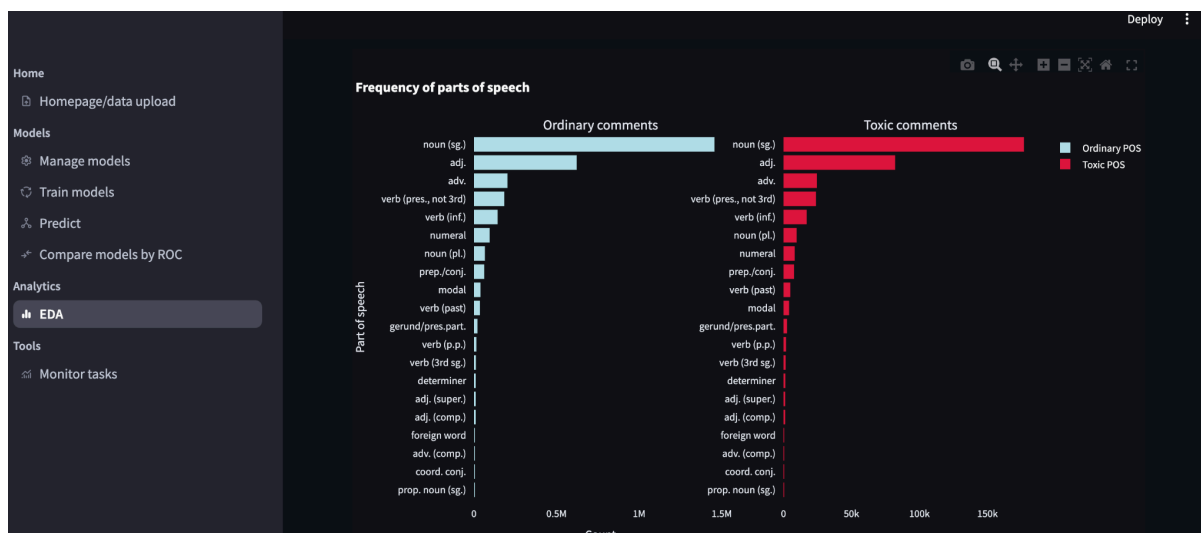
|   | uuid                                 | name  | status  | result_msg | updated_at                 |
|---|--------------------------------------|---|---------|------------|----------------------------|
| 0 | 263d5dfe-2013-4cd7-8925-df45a976b9dd | Обучение модели<br>'logistic_regression_57dc1a99' | running | <NA>       | 2024-12-30T06:57:46.684579 |

Во вкладке **Manage models** вы можете управлять текущими моделями(удалять модель, загружать инференс и выгружать инференс модели, удалять все модели)

| Model ID                        | Model type              | LOAD/UNLOAD | DELETE |
|---------------------------------|-------------------------|-------------|--------|
| default_logistic_regression     | logistic_regression     | Unload      | Delete |
| default_linear_svc              | linear_svc              | Unload      | Delete |
| default_multinomial_naive_bayes | multinomial_naive_bayes | Unload      | Delete |
| logistic_regression_57dc1a99    | logistic_regression     | Load        | Delete |

Во вкладке **EDA** вы можете получить аналитику по ранее загруженному csv - иногда возможна долгая обработка и подготовка аналитики





Во вкладке **Predict** вы можете получить предсказанное значение токсичности комментария если передадите текст в textarea - очень важно передавать строку без переносов в textarea иначе она будет считана некорректно

The 'Predict toxicity' interface shows a dropdown menu for selecting a model (currently 'default\_logistic\_regression'). Below it is a text area for inputting text, with the example 'sd' entered. A red button labeled 'Obtain predictions' is positioned below the text area. The 'Predictions' section contains a table with the following data:

|   | Text | Predictions |
|---|------|-------------|
| 0 | sd   | Not toxic   |

Во вкладке **Compare models by ROC** вы можете выбрать одну или несколько моделей для построение метрик и выбора наилучшей модели - в рамках интерфейса вы можете использовать как существующий файл(который ранее был загружен в главной вкладке) выбрав галочку **Use previous (train) data** так и загрузить свой собственный (убрав эту галочку) для построения графиков нажмите кнопку "Построить графики"

