report.md

Техническое описание проекта

Реализация четвертого чекпоинта находится в директории ./app

Техническое описание будет опираться на требования, указанные в файле "Чекпоинт 4. Сервис."

https://docs.google.com/presentation/d/16WZAKfUzXpo_ojw2OhCNg25aVSKCAPVVmYdnDGmA1yQ/edit#slide=id.p

Ведение разработки в репозитории

- Вся разработка приложения велась в отдельной ветке `app_backend`. Я не создавал отдельных веток для /backend и /frontend частей по той причине, что код писался мною в одиночку и наличие нескольких веток усложнило бы для меня процесс разработки.

Качество кода

- По согласованию с куратором все .py-файлы были пропущены через `black` и успешно прошли проверку стиля `flake8`.

Реализация серверной части

Реализация серверной части находится в директории ./backend. Все эндпоинты реализованы в `main.py`.

Функции для обучения нейронной сети (далее НС) и проведения EDA реализованы в `model_preprocessing.py`

Описание реализованных ручек:

- `POST /fit` - обучение НС на основе изображений, подаваемых клиентом на вход модели.

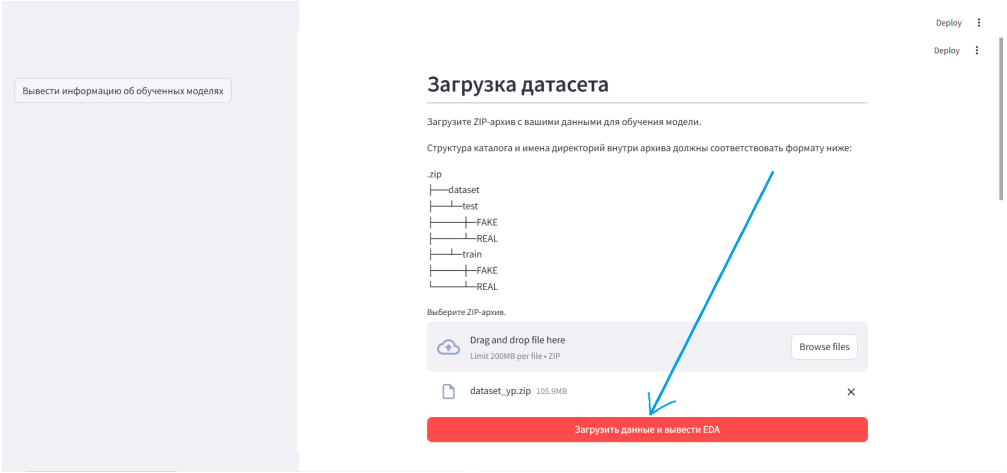
  В качестве сверточного блока НС используется предобученный сверточный блок `resnet18`. Признаки на выходе из сверточного блока подаются на вход в линейный слой (который состоит из одного нейрона) и далее подаются в функцию активации (сигмоиду). В качестве функции ошибок используется log-loss. Т.е. это логистическая регрессия над вектором признаков из сверточного блока. Такое решение обусловлено условиями третьего чекпоинта, где обязательным требованием было создать линейную модель. А в этом чекпоинте обязательным требованием было использовать модель с предыдущего чекпоинта.

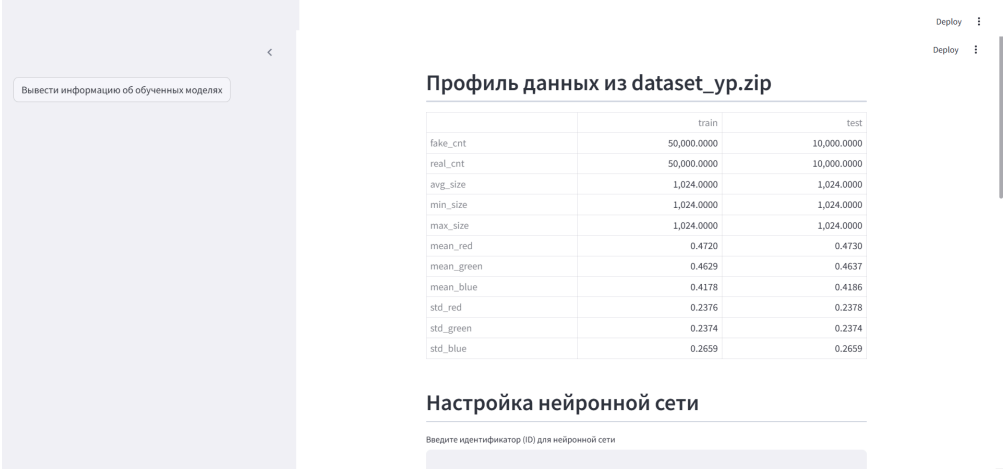
- `POST /set` - установка модели (среди, тех что были обучены через `/fit`) в качестве модели для инференса.
- `POST /predict` - предсказание метки класса (REAL/FAKE) для изображения моделью для инференса.
- `GET /models` - вывод списка обученных моделей с подробной информацией о них. В эту информацию входят значения гиперпараметров, accuracy, вид предобученной модели, а также функция активации на последнем линейном слое.
- `GET /eda` - расчет профиля данных, подаваемых клиентом на вход модели. Профиль данных состоит из следующих статистик в разрезе `train/test` выборок:
  - `fake_cnt` - количество фейковых изображений;
  - `real_cnt` - количество реальных изображений;
  - `avg_size` - средний размер изображения в выборке (указывается среднее число пикселей на изображении);
  - `min_size` - минимальный размер изображения в выборке;
  - `max_size` - максимальный размер изображения в выборке;
  - `mean_red` - среднее значение по красному каналу;

- `mean_green` - среднее значение по зеленому каналу;
- `mean_blue` - среднее значение по синему каналу;
- `std_red` - стандартное отклонение по красному каналу;
- `std_green` - стандартное отклонение по зеленому каналу;
- `std_blue` - стандартное отклонение по синему каналу;

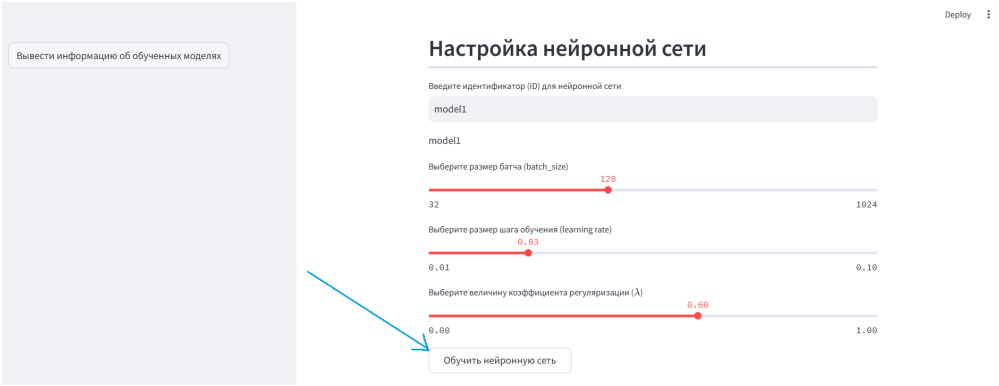
Реализация сервиса на Streamlit

1. Необходимо загрузить архив с изображениями для обучения в определенном формате который указан в приложении (для удобства тестирования архив с данными в нужном формате дополнительно размещен на Яндекс Диске) (https://disk.yandex.ru/d/Kf_7F4yervF1pQ). После того как вы выберете архив и нажмете на кнопку "Загрузить данные и вывести EDA", архив будет загружен и распакован в специально отведенной директории и начнется расчет профиля данных. Весь процесс после нажатия кнопки до отображения профиля занимает ~5 минут (это если брать датасет с Яндекс Диска на 120к изображений).





2. После того, как вы загрузите архив с данными, вы можете приступить к настройке гиперпараметров нейронной сети. Для этого обязательно нужно ввести идентификатор для нового экземпляра НС. Далее необходимо нажать на кнопку "Обучить нейронную сеть". В течение 30 секунд НС будет обучена на ваших данных (На локальном компьютере обучение занимает ~7 секунд, т.к. у меня GPU. В докере по умолчанию используется CPU).

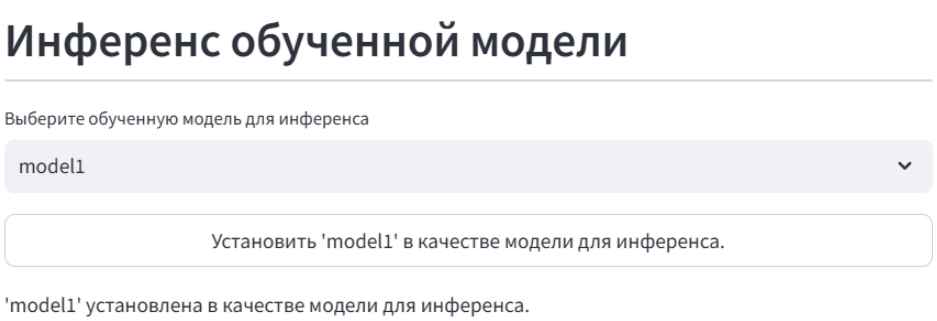


3. После того, как вы обучили несколько моделей, вы можете вывести всю информацию об этих моделях, нажав на кнопку "Вывести информацию об обученных моделях" в левом верхнем углу.

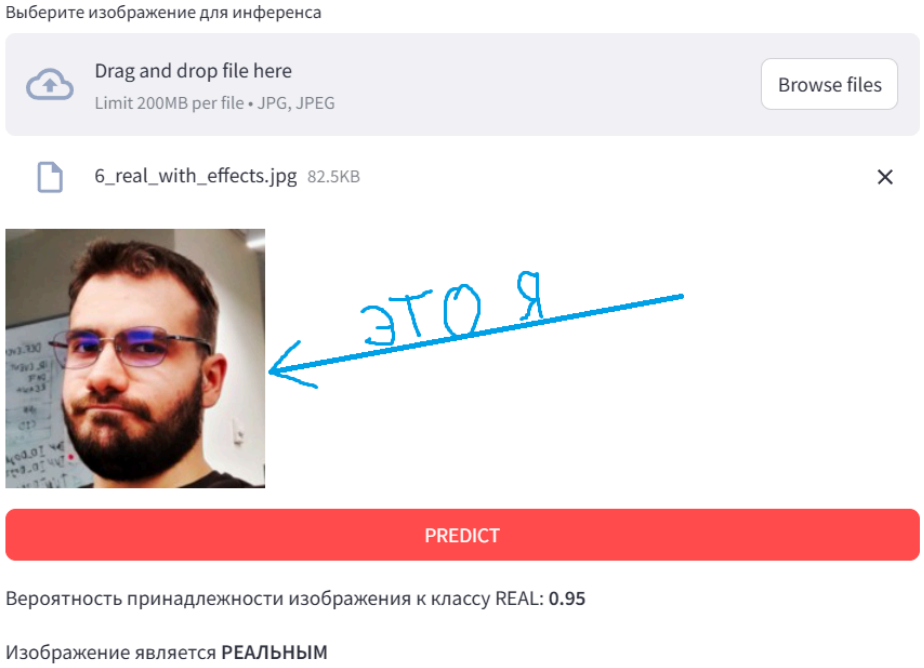
Вывести информацию об обученных моделях


```
{
  "model1" : {
    "type_nn_pretrain" : "ResNet18"
    "end_activation_function" : "Sigmoid"
    "batch_size" : 128
    "lr" : 0.03
    "C" : 0.1
    "num_epochs" : 1
    "accuracy" : 0.7452
  }
  "model2" : {
    "type_nn_pretrain" : "ResNet18"
    "end_activation_function" : "Sigmoid"
    "batch_size" : 128
    "lr" : 0.01
    "C" : 0.26
    "num_epochs" : 1
    "accuracy" : 0.7472
  }
```


4. Далее необходимо выбрать модель для инференса и нажать на кнопку "Установить модель в качестве инференса". Для выбора будут доступны только уже обученные вами модели.



5. После того, как вы установите модель для инференса, вам станет доступна загрузка изображения для инференса. Выберите любое изображение в формате `.jpg` (это важно!) и нажмите кнопку `PREDICT`. Будут выведены вероятность принадлежности объекта к тому или иному классу, а также вердикт модели (в модели стоит treshold >= 0.60).



Создание docker-приложения

1. Необходимо зайти в директорию `./app` и в командной строке ввести команду `docker compose up --build`. Первоначальная сборка занимает ~10 минут.
2. Далее в Docker Desktop можно зайти в композицию контейнеров и перейти непосредственно в само приложение `frontend`.