

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Факультет информатики
Кафедра технической кибернетики

Отчет по лабораторной работе №2

Дисциплина: «Инженерия данных»

Тема: «Инференс и обучение НС»

Выполнил: Мелешенко И.С.

Группа: 6233-010402D

Самара 2023

СОДЕРЖАНИЕ

Часть 1. Построение пайплайн для инференса данных.	3
Шаг 1. Разработка и реализация DAG-а	3
Шаг 2. Регистрация на huggingface и получения токена API.	4
Шаг 3. Создание Docker образа с необходимыми библиотеками.	5
Шаг 4. Подготовка DAG-а.	7
Шаг 5. Запуск DAG-а.	8
Часть 2. Пайплайн, который реализует систему автоматического обучения/дообучения нейросетевой модели	11
Шаг 1. Разработка DAG.....	11
Шаг 2. Запуска DAG-а.	12
Заключение.....	13

Часть 1. Построение пайплайн для инференса данных.

Шаг 1. Разработка и реализация DAG-а

В рамках первого задания необходимо реализовать пайплайн, который реализует систему "Автоматического распознавания речи" для видеофайлов.

Построенный пайплайн будет выполнять следующие действия поочередно:

- Производить мониторинг целевой папки на предмет появления новых видеофайлов.
- Извлекать аудиодорожку из исходного видеофайла.
- Преобразовывать аудиодорожку в текст с помощью нейросетевой модели.
- Формировать конспект на основе полученного текста.
- Формировать выходной .pdf файл с конспектом.

Для реализации описанных действий мы будем использовать DockerOperator, а также FileSensor для получения необходимого видеофайла.

Для работы task-а по ожиданию получения нового видео необходимо создать новое подключение к airflow. Для создания подключения переходим в Airflow по адресу <http://localhost:8080/connection/list/> или мы можем в Airflow пройти по пути Admin>>Connections, как на рисунке ниже.

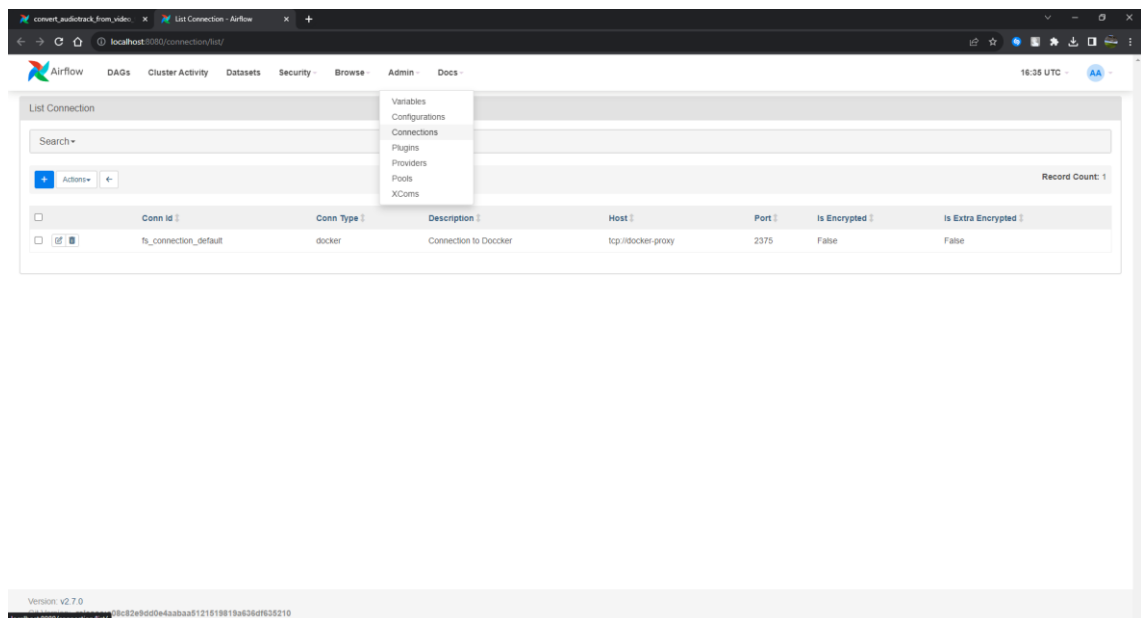


Рисунок 1 – Создание Connection

Connection Id *

Connection Type *

Description

Registry URL

Username

Password

Port

Extra

Рисунок 2 – Параметры Connection

Шаг 2. Регистрация на huggingface и получения токена API.

Далее для того, чтобы можно было преобразовать наш аудиофайл в текст, а после получить из него summary, необходимо зарегистрироваться на <https://huggingface.co/> и получить токен API с правами записи для возможности отправки и получения запросов к сайту.

Join Hugging Face

Join the community of machine learners!

Email Address

Password

Next

Already have an account? [Log in](#)

SSO is available for companies

Рисунок 3 – Регистрация на huggingface

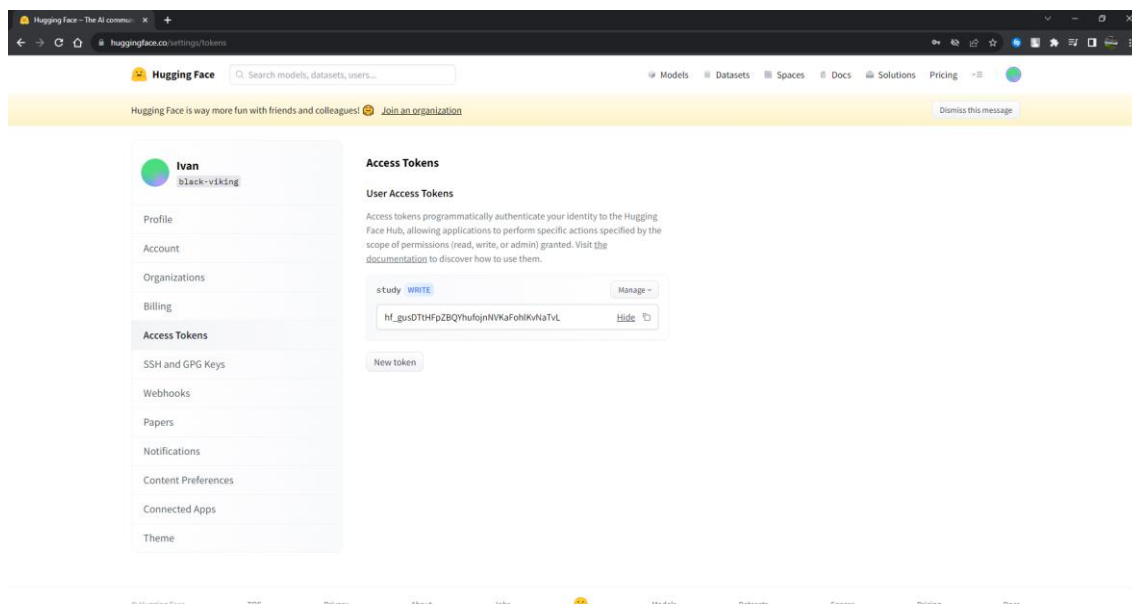


Рисунок 4 – Получение токена API

Шаг 3. Создание Docker образа с необходимыми библиотеками.

Для сохранения конспекта в PDF, необходимо было использовать библиотеку `fpdf`. Создадим необходимый для этого образ в Docker, который будет содержать в себе необходимые библиотеки для выполнения всей лабораторной работы. Процесс представлен ниже.

Вначале создадим Dockerfile который будет содержать в себе инструкции по сборке и разворачиванию контейнера. Контейнер мы создаем на основе `tensorflow`, который нам пригодится при выполнении второй части работы. Так же добавим следующие библиотеки, которые нам понадобятся в будущем:

- Scikit-learn
- Numpy
- Pandas
- FPDF

Пример Dockerfile, который я использовала в лабораторной работе приведен на рисунке ниже, а также в репозитории с решением лабораторной работы. После того как Dockerfile создан переходим в консоль и выполним процесс сборки.

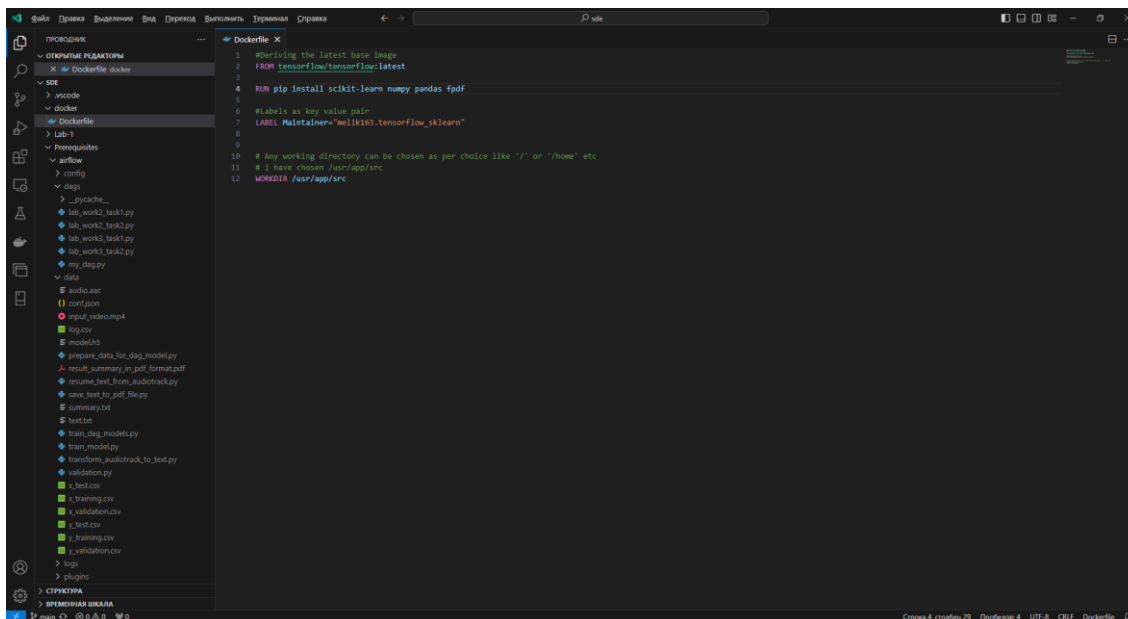


Рисунок 7 – Создание Dockerfile

Первым делом произведем сборку образа, при помощи команды:

```
$ docker build . -t our_tensorflow_container
```

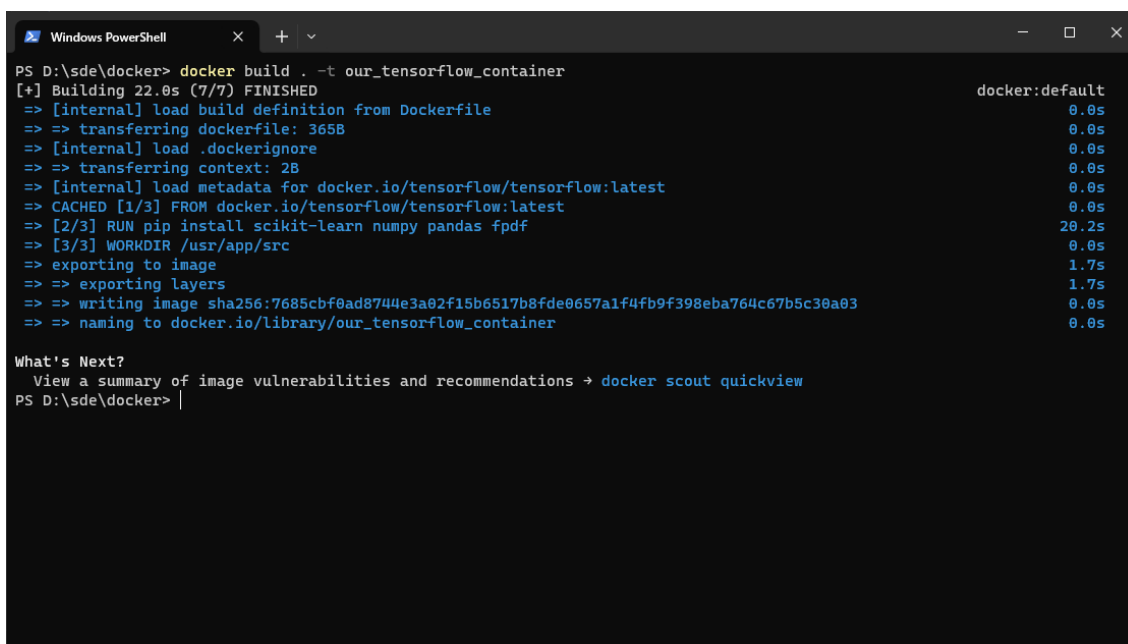


Рисунок 8 – Сборка образа

После успешной сборки, необходимо произвести проверку того, что наш Docker образ был создан. Для этого используем команду

```
$ docker images
```

После проверки произведем присвоение tag нашему образу, для того чтобы можно было произвести отправку нашего контейнера в DockerHub. Для этого воспользуемся командой:

```
$ docker tag -t our_tensorflow_container melik163/our_tensorflow_container:1.0
```

```
Windows PowerShell
> exporting to image 1.7s
> exporting layers 1.7s
> writing image sha256:7685cbf0ad8744e3a02f15b6517b8fde0657a1f4fb9f398eba764c67b5c30a03 0.0s
> naming to docker.io/library/our_tensorflow_container 0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview

PS D:\sde\docker> docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
our_tensorflow_container  latest     7685cbf0ad87  About a minute ago  2.1GB
mlflow-web          latest     da035e44318c  4 days ago    765MB
airflow-airflow-triggerer latest     4df00f2bf0d5  4 days ago    1.82GB
airflow-airflow-worker latest     3d8f4b21596c  4 days ago    1.82GB
airflow-airflow-init latest     01d790f408e0  4 days ago    1.82GB
airflow-airflow-webserver latest     31c323886c61  4 days ago    1.82GB
airflow-airflow-scheduler latest     0a66fdf03f9a  4 days ago    1.82GB
tensorflow/tensorflow latest     6a8c4ad355be  11 days ago    1.78GB
minio/minio         latest     88c665b1183a  12 days ago    147MB
minio/mc             latest     eaa326464fd5  12 days ago    76.9MB
sasha151299/my_pdf  1.0       0a6eaffde1ba  3 weeks ago    1.12GB
redis                latest     961dda256baa  3 weeks ago    138MB
postgres            15        8cde386e2e85  3 weeks ago    419MB
postgres            13        19975f71ce75  3 weeks ago    413MB
docker              24-dind   daefcf9ccf3b  5 weeks ago    336MB
apache/nifi         1.23.2    81455911cd05  3 months ago    1.94GB
dpkg/pgadmin4       7.6       881febbc9e93  3 months ago    534MB
nshou/elasticsearch-kibana kibana7    17f031ca3406  7 months ago    1.18GB
mysql/mysql-server  5.7.28    c8c8ef4f3c81  4 years ago     310MB

PS D:\sde\docker> docker tag our_tensorflow_container melik163/our_tensorflow_container:1.0
PS D:\sde\docker> |
```

Рисунок 9 – Присвоение тега образу

В итоге после всех приготовлений, произведем отправку нашего образа в DockerHub, при помощи команды:

```
$ docker push melik163/our_tensorflow_container:1.0
```

```
Windows PowerShell

tensorflow/tensorflow latest 6a8c4ad355be 11 days ago 1.78GB
minio/minio          latest 88c665b1183a 12 days ago 147MB
minio/mc              latest eaa326464fd5 12 days ago 76.9MB
sasha151299/my_pdf   1.0    0a6eaffde1ba 3 weeks ago 1.12GB
redis                latest 961dda256baa 3 weeks ago 138MB
postgres            15     8cde386e2e85 3 weeks ago 419MB
postgres            13     19975f71ce75 3 weeks ago 413MB
docker              24-dind daefcf9ccf3b 5 weeks ago 336MB
apache/nifi         1.23.2 81455911cd05 3 months ago 1.94GB
dpkg/pgadmin4       7.6     881febbc9e93 3 months ago 534MB
nshou/elasticsearch-kibana kibana7 17f031ca3406 7 months ago 1.18GB
mysql/mysql-server  5.7.28 c8c8ef4f3c81 4 years ago 310MB

PS D:\sde\docker> docker push melik163/our_tensorflow_container:1.0
The push refers to repository [docker.io/melik163/our_tensorflow_container]
e1f360959148: Pushed
06e66be4628b: Pushed
75acb1242fe3: Mounted from tensorflow/tensorflow
1d7a2a211a6b: Mounted from tensorflow/tensorflow
2db699de670e: Mounted from tensorflow/tensorflow
0cf31f98a4b6: Mounted from tensorflow/tensorflow
f663f4c9c5b6: Mounted from tensorflow/tensorflow
104e4c35057a: Mounted from tensorflow/tensorflow
eb864c00a034: Mounted from tensorflow/tensorflow
94235a128255: Mounted from tensorflow/tensorflow
0ac81db158f3: Mounted from tensorflow/tensorflow
8e8c3d39273b: Mounted from tensorflow/tensorflow
f99aba8580cb: Mounted from tensorflow/tensorflow
256d88da4185: Mounted from tensorflow/tensorflow
1.0: digest: sha256:2400063992556f7cc9612f9eb0feb9d0589f6656eb255a6ff179878d422e3737 size: 3250
PS D:\sde\docker> |
```

Рисунок 10 – Отправка образа в DockerHub

Шаг 4. Подготовка DAG-a.

В результате выполнения данной части работы был разработан DAG, состоящий из 5 task-ов:

`wait_get_new_videofile` – осуществляет «прослушивание» указанной директории, на предмет появления в ней видеофайла, который будет принят далее в работу.

`extract_audiotrack_from_video` – осуществляет извлечение аудиодорожки из исходного видеофайла для дальнейшей работы. Для извлечения аудиодорожки из видео была использована библиотека `ffmpeg`, которая была получена из Docker-образа `jrottenberg/ffmpeg`.

`transform_audiotrack_to_text` – осуществляет обработку, распознавание и трансформацию аудиофайла в текстовый файл. Данная операция осуществлялась при помощи запросов в сервис `huggingface`.

`resume_text_from_audiotrack` – осуществляет суммаризацию текстового файла, который получен на предыдущих этапах.

`save_get_text_from_txt_to_pdf` – осуществляет сохранение полученного результата в файл формата `pdf`.

Шаг 5. Запуск DAG-а.

Теперь после всех необходимых настроек и приготовлений, мы можем запустить наш DAG. Для этого переходим в `airflow`: <http://localhost:8080/home> и находим наш только что созданный DAG:

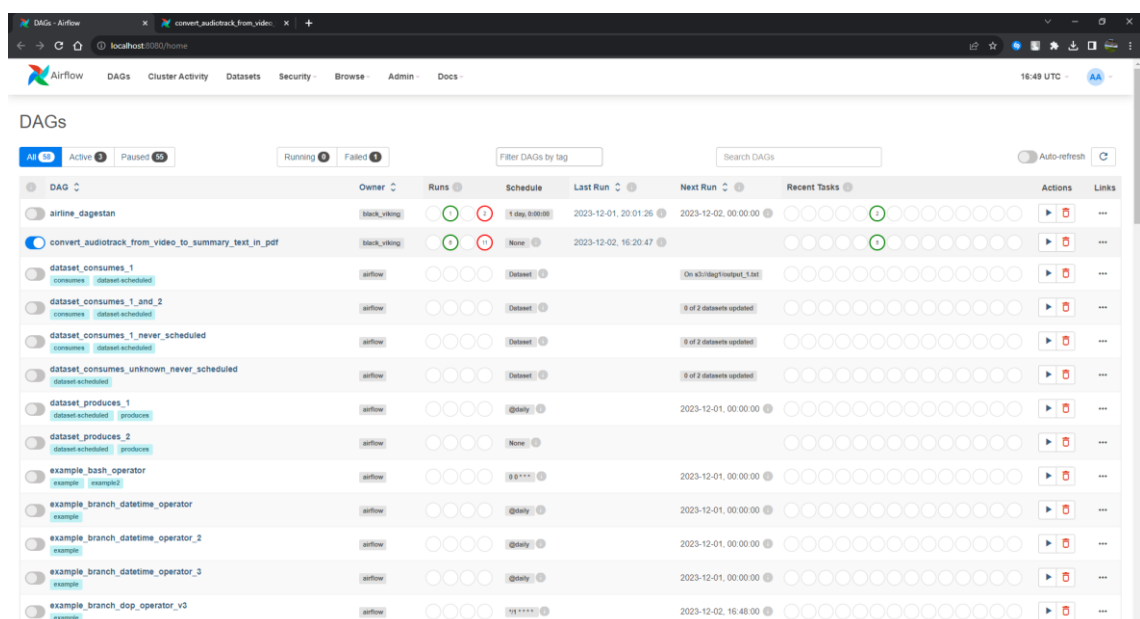


Рисунок 5 – Поиск DAG-а.

Далее запускаем наш DAG и наслаждаемся процессом.

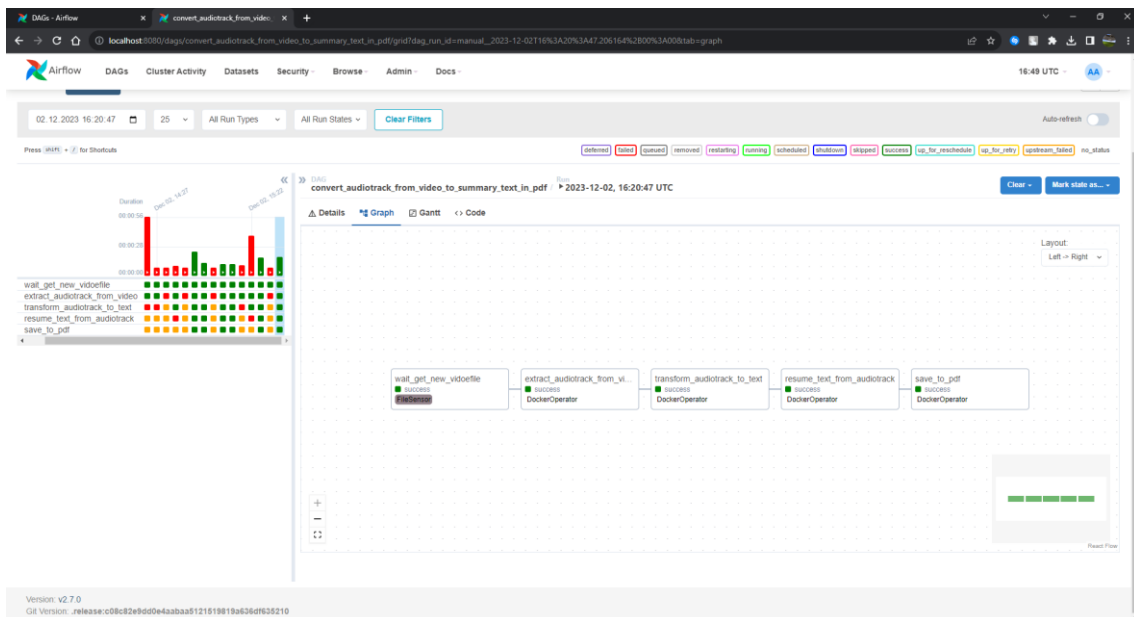


Рисунок 6 – Запуск DAG-а.

В качестве исходного видео использовался фрагмент из кинофильма «Крестный отец» длительностью 3 минуты 11 секунд. После чего мы получали аудиодорожку, которая использовалась в качестве основы для получения текстового файла.



Рисунок 11 – Результат работы huggingface по преобразованию аудио в текст

Далее полученный результат мы еще раз передавали huggingface для получения уже конспекта по отправленному нами файлу. Полученный результат записывали pdf-файл.

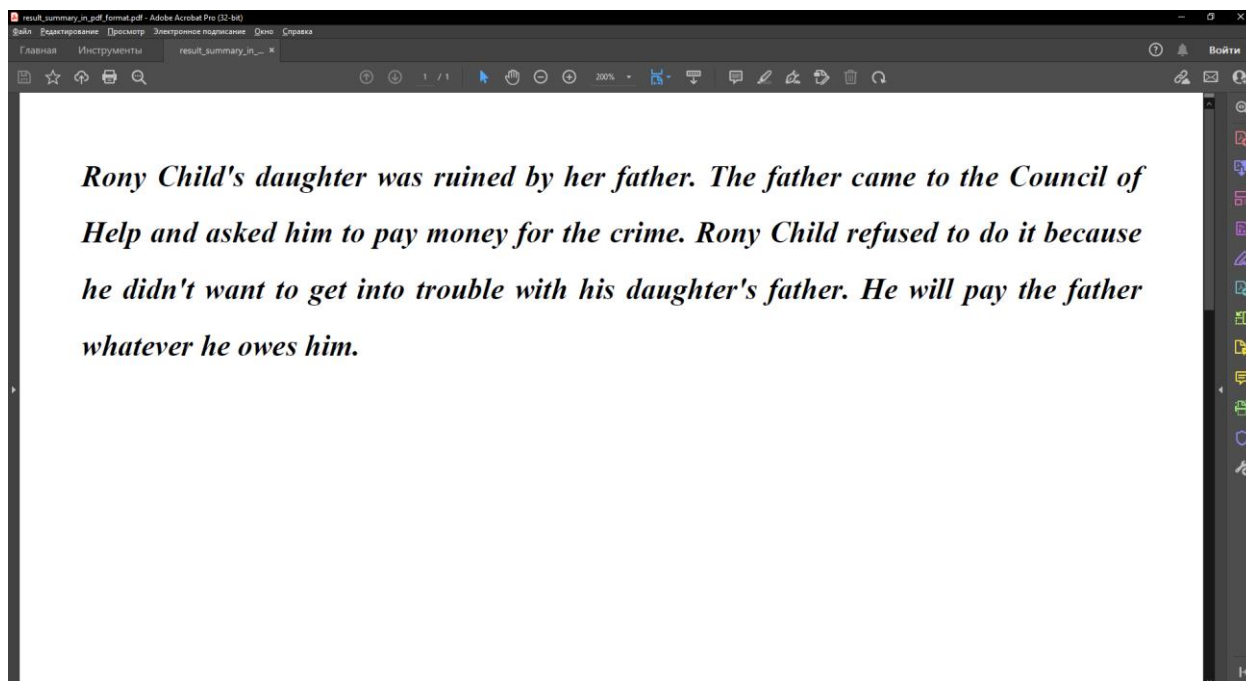


Рисунок 12 – Конспект текстового файла.

Получилось неплохо. Перейдем ко второй части.

Часть 2. Пайплайн, который реализует систему автоматического обучения/дообучения нейросетевой модели

В рамках второй части лабораторной работы нам необходимо было разработать пайплайн, который реализует систему автоматического обучения/дообучения нейросетевой модели.

Шаг 1. Разработка DAG

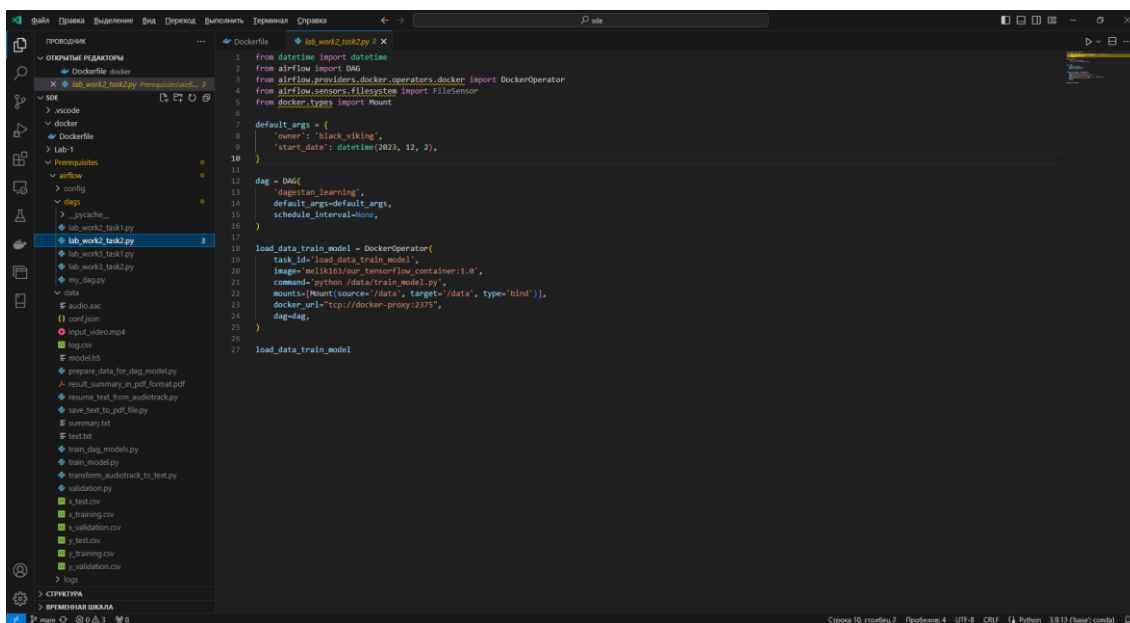


Рисунок 13 - Пайплайн

DAG запускал код, который получал датасет вин `load_wine` из `sklearn.datasets`, после чего мы проводили разбиение данных. Которые передаются в нейросеть, после чего модель проходит обучение. Процесс обучения логируется.

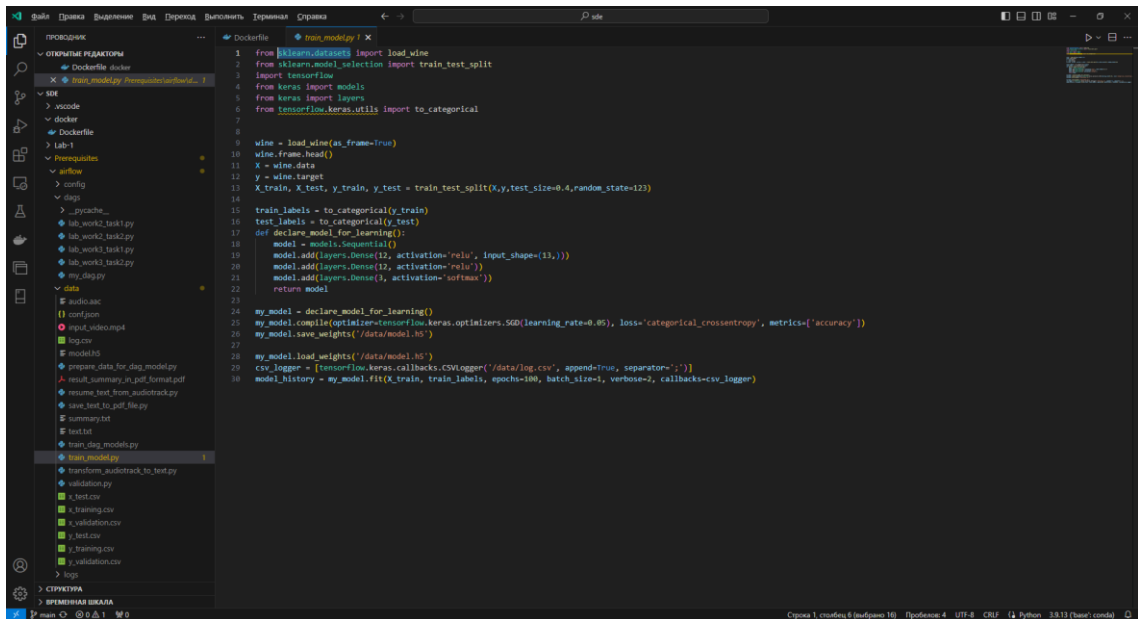


Рисунок 14 – Код обучения модели.

Шаг 2. Запуска DAG-а.

В процессе запуска DAG-а модель была обучена и показала какие-то результаты, которые мы записали в файл. В итоге получили вот такой лог обучения:

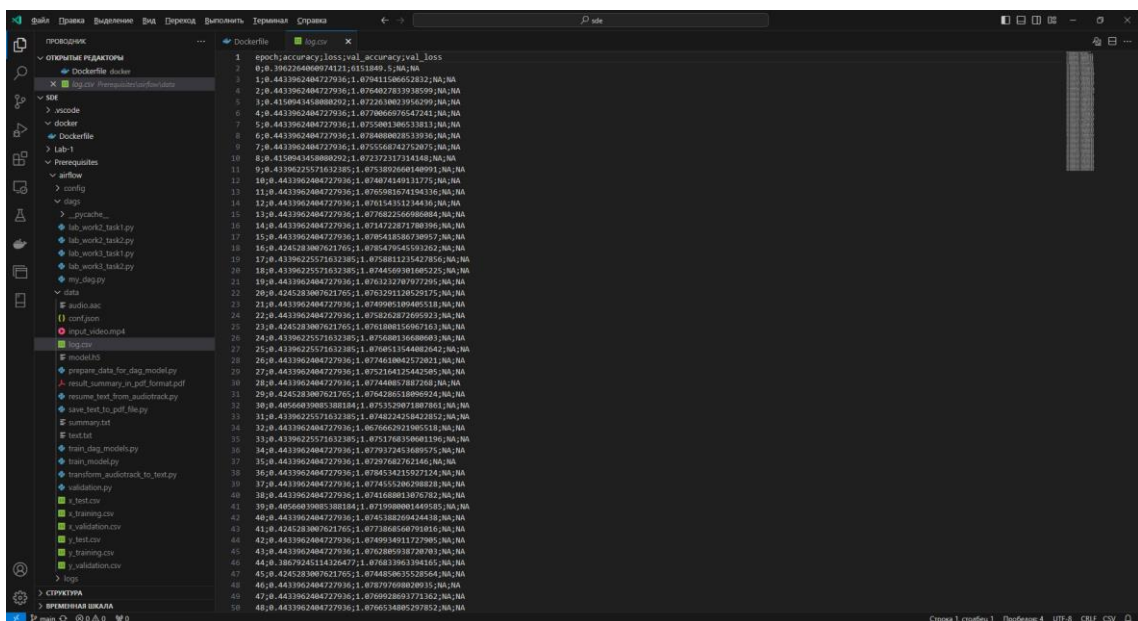


Рисунок 15 – Лог процесса обучения нейросети.

Заключение

В заключении хотелось бы отметить полезные навыки, полученные в результате выполнения лабораторной работы:

1. Работа с DAG в Airflow
2. Работе сетями на huggingface