

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  
Кафедра технической кибернетики

**Отчет по лабораторной работе №2**

Дисциплина: «Инженерия данных»

Тема: «**Инференс и обучение НС**»

Выполнил: Дубман Л.Б.

Группа: 6233-010402D

Самара 2024

## **Задание на лабораторную работу**

### ***Пайплайн для инференса данных***

В рамках данного задания предлагается построить пайплайн, который реализует систему "Автоматического распознавания речи" для видеофайлов.

Построенный пайплайн будет выполнять следующие действия поочередно:

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

### ***Пайплайн для обучения модели***

В рамках данного задания предлагается построить пайплайн, который реализует систему автоматического обучения/дообучения нейросетевой модели.

Предлагается самостоятельно выбрать набор данных и модель для обучения. Например, можно реализовать пайплайн для обучения модели, которую вы планируете использовать в вашей НИР или ВКРМ. Это также позволит вам добавить отдельный пункт в ваш отчет.

Итак, пайплайн будет выполнять следующие действия:

1. Читать набор файлов из определенного источника (файловой системы, сетевого интерфейса и т.д.).
2. Формировать пакет данных для обучения модели.
3. Обучать модель.

4. Сохранять данные результатов обучения (логи, значения функции ошибки) в текстовый файл

Для успешного выполнения задания необходимо продемонстрировать успешность обучения модели и приложить файл `.irunb`, в котором продемонстрирован процесс инференса данной модели.

## СОДЕРЖАНИЕ

Часть 1. Построение пайплайн для инференса данных. ....	5
Шаг 1. Разработка и реализация DAG-а .....	5
Шаг 2. Регистрация на huggingface и получения токена API.....	7
Шаг 3. Создание Docker образа с необходимыми библиотеками. ....	9
Шаг 4. Подготовка DAG-а. ....	12
Шаг 5. Запуск DAG-а. ....	13
Часть 2. Пайплайн, который реализует систему автоматического обучения/дообучения нейросетевой модели.....	16
Шаг 1. Разработка DAG .....	16
Шаг 2. Запуска DAG-а. ....	18
Заключение .....	20

## 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.

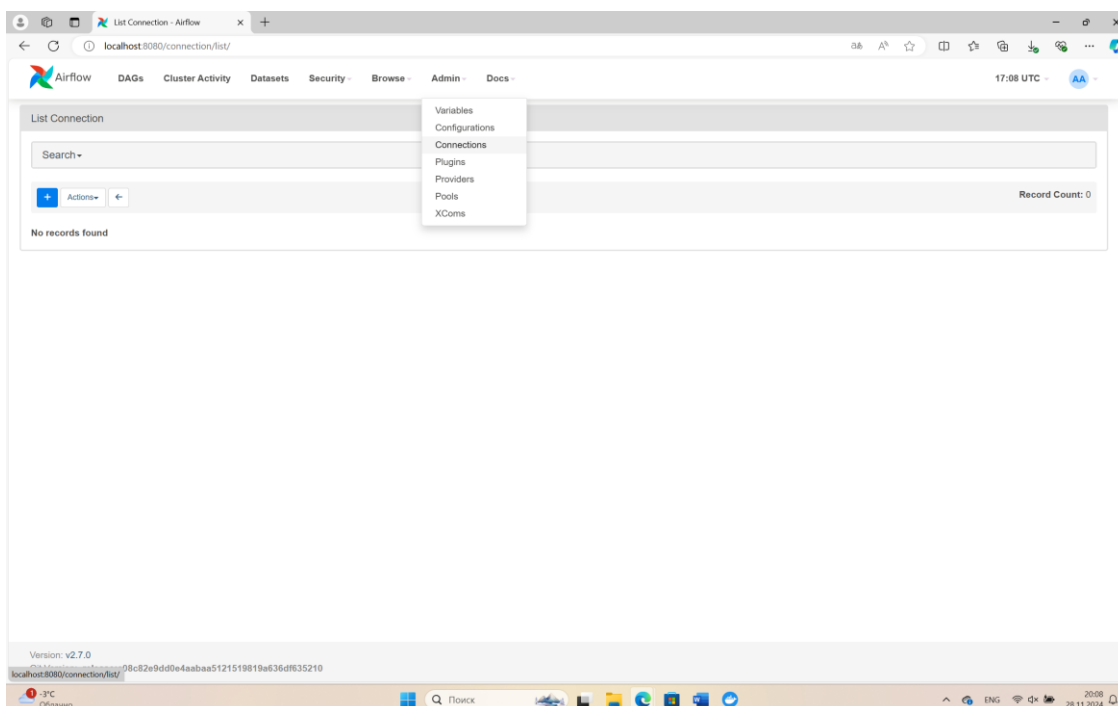


Рисунок 1 – Создание Connection

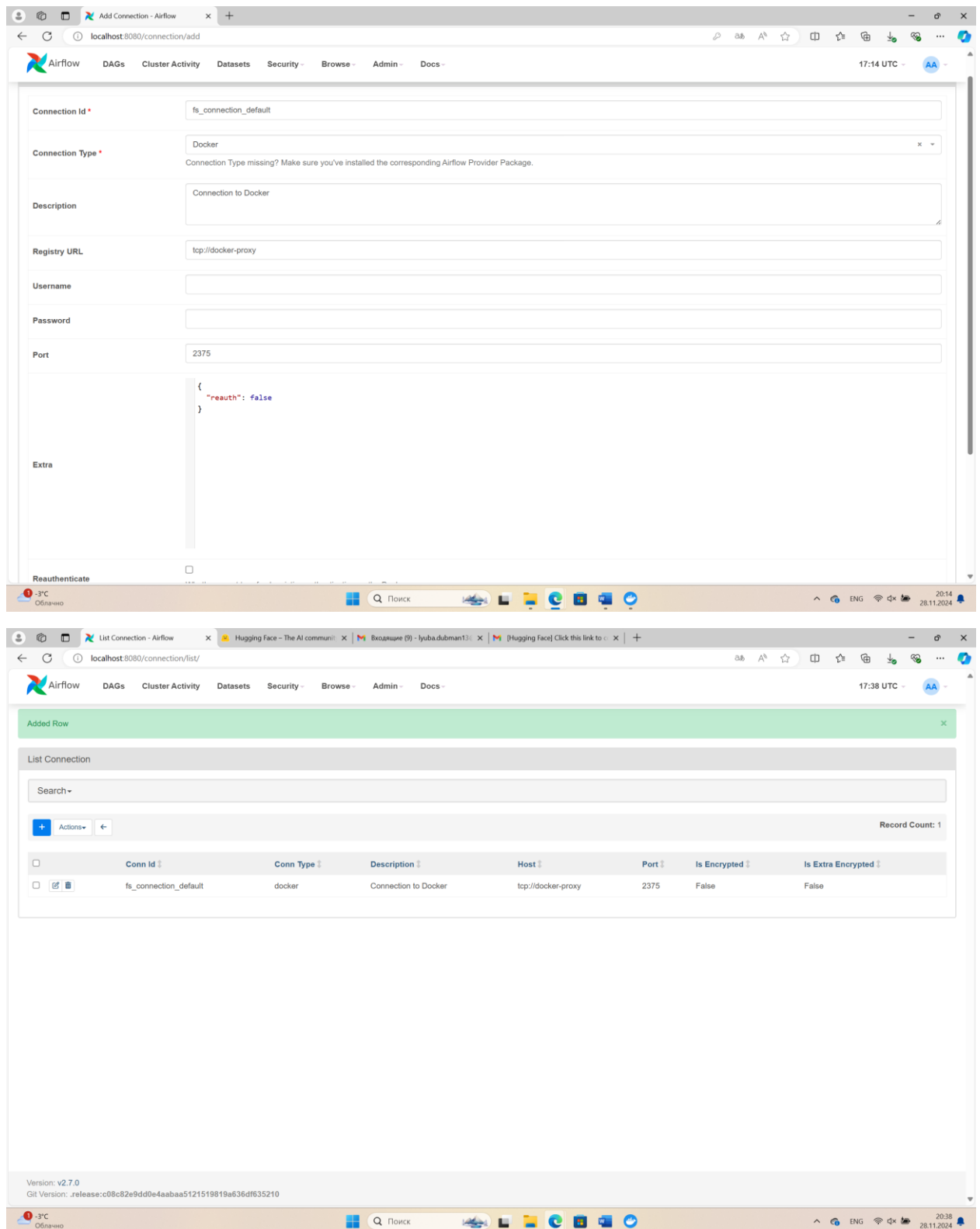


Рисунок 2 – Параметры Connection

## 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.

The screenshot displays the Hugging Face registration interface. At the top, there's a navigation bar with links for Models, Datasets, Spaces, Posts, Docs, Enterprise, Pricing, Log In, and Sign Up. The main content area features a 'Join Hugging Face' card with a yellow robot icon. The card prompts users to 'Join the community of machine learners!' and provides a form for registration. The 'Email Address' field is filled with 'hawonmin1997@gmail.com'. Below it, a hint suggests using an organization email. The 'Password' field is masked with dots and includes a toggle for visibility. Three checkmarks indicate password requirements: 'Must contain at least 8 characters', 'Must contain uppercase, lowercase letters, and numbers', and 'If less than 12 characters, must contain a special character'. A 'Next' button is positioned below the password field. At the bottom of the card, there's a link for 'Already have an account? Log in'. The footer of the page shows 'SSO is available for Enterprise accounts.' The Windows taskbar at the bottom indicates a temperature of -3°C, a search bar, and the date 28.11.2024 at 20:15.

Рисунок 3 – Регистрация на *huggingface*

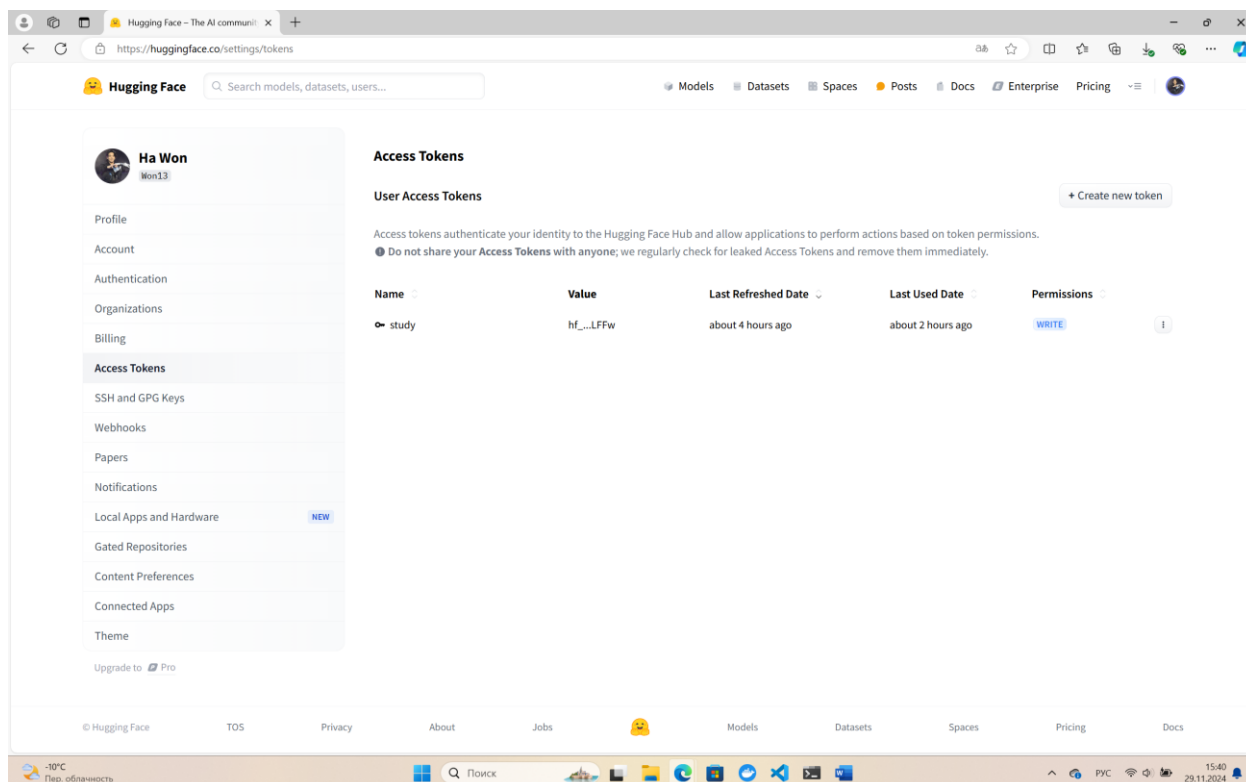


Рисунок 4 – Получение токена API

API токен = hf\_vXMQmtxvHGoZnllVbsKkWQPFCQYbXdLFFw



### 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.

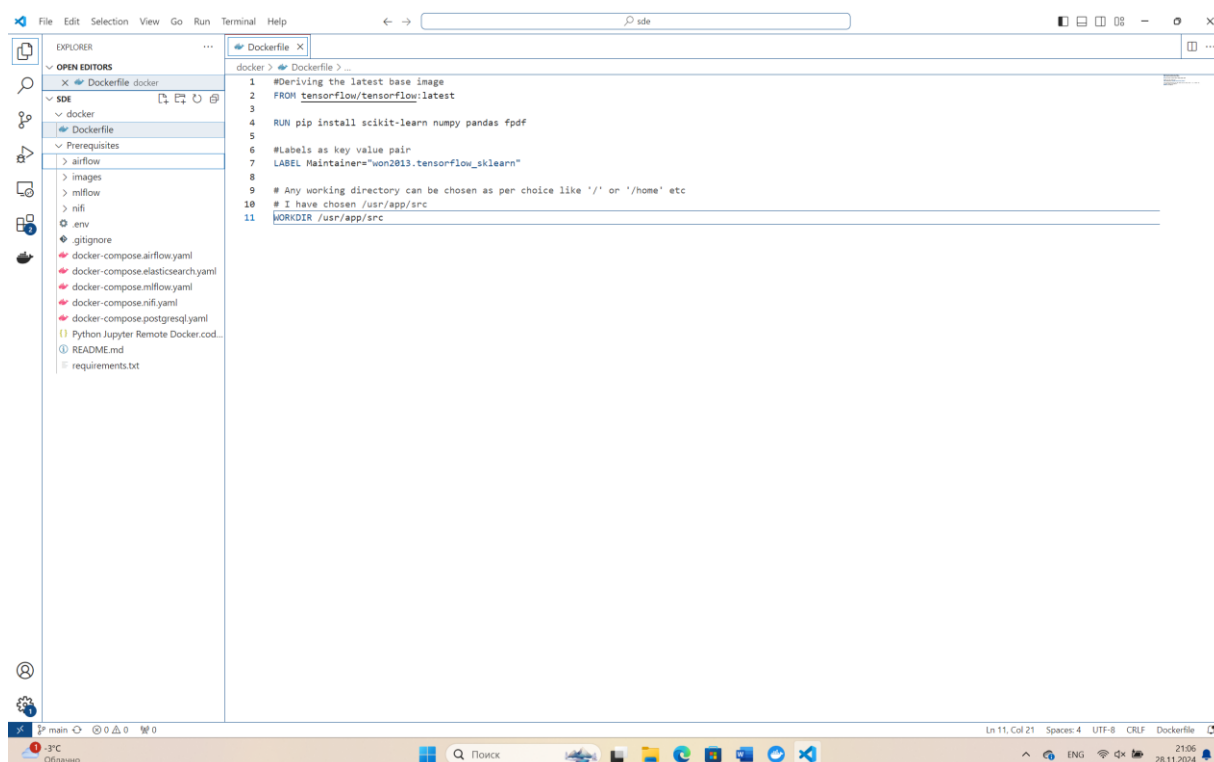


Рисунок 7 – Создание Dockerfile

Первым делом произведем сборку образа, при помощи команды:

\$ docker build . -t our\_tensorflow\_container

```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Установите последние версии PowerShell для новых функций и улучшения! https://aka.ms/PSWindows

PS C:\Users\lyuba> cd data_engineering\sde\docker
PS C:\Users\lyuba\data_engineering\sde\docker> docker build . -t our_tensorflow_container
[+] Building 97.4s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 362B
=> [internal] load metadata for docker.io/tensorflow/tensorflow:latest
=> [auth] tensorflow/tensorflow:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/tensorflow/tensorflow:latest@sha256:f3be5db24f080b3b228a23eb24f586058b3bec445d81500f9516
=> => resolve docker.io/tensorflow/tensorflow:latest@sha256:f3be5db24f080b3b228a23eb24f586058b3bec445d81500f9516
=> => sha256:cd877f681d4feb5f423c9014e161cf85e9f5bb37de840f7d9e62b1b0645b31c86 858B / 858B
=> => sha256:7594693ef1619703fa59c49d49f1ce3c9f8dd4161d1d2ef162b5cf7efb4edd731 4.53kB / 4.53kB
=> => sha256:6411378b647700feebf4903d4b951d134a1947ce092d080deb23a544b3684ac 29.54MB / 29.54MB
=> => sha256:70464040456b253f3544f76746d197434778dacb313e26cc07f2d70d45fedc39 766B / 766B
=> => sha256:f3be5db24f080b3b228a23eb24f586058b3bec445d81500f95167c70e5473d4e 2.83kB / 2.83kB
=> => sha256:2443cd0be7bb6a9c615e32e3c44d3a3e8bcab2f73190721c08a033a10ca2e797 164B / 164B
=> => sha256:0b3e4bf45acf2ea793433a225dcfdcab61562eaa52abee05f7d9794f5dbddef 44.28MB / 44.28MB
=> => sha256:cbdf062747acb4a231665a488acc0aef0cd1cdac9ab0eda6f73f113b7424b15 133.29MB / 133.29MB
=> => sha256:cf450bbc2f912c0a2236789627065e3e96b53d51763e7182653557d43ba98ba7 960B / 960B
=> => extracting sha256:6411378b647700feebf4903d4b951d134a1947ce092d080deb23a544b3684ac 1.77s
=> => sha256:a886dd517e77346add853708010a931c5d5b220c30f9b84125e253e69858be34 127B / 127B
=> => sha256:a9eb4219fce39e31343dd629ad7697f6a46e41e25ff4933709d7bf5e41bf685 37.41MB / 37.41MB
=> => extracting sha256:d877f681d4feb5f423c9014e161cf85e9f5bb37de840f7d9e62b1b0645b31c86 0.0s
=> => extracting sha256:70464040456b253f3544f76746d197434778dacb313e26cc07f2d70d45fedc39 0.0s
=> => extracting sha256:2443cd0be7bb6a9c615e32e3c44d3a3e8bcab2f73190721c08a033a10ca2e797 0.0s
=> => sha256:9cc72bd2d4f50a0e775691c7198a1af3919a9b96f39b86cb44bd8aa5f946d 330.31MB / 330.31MB
=> => sha256:6a0550e4123ec5614f97f69abb5ca767c1c43b213a52af3087ab15bba018ae9c 1.13kB / 1.13kB
=> => extracting sha256:0b3e4bf45acf2ea793433a225dcfdcab61562eaa52abee05f7d9794f5dbddef 0.6s
=> => sha256:3a6e466541337435bc0d7230b38695810ab694d60dcca510462a5e187e5d52c 1.13kB / 1.13kB
=> => extracting sha256:cbdf062747acb4a231665a488acc0aef0cd1cdac9ab0eda6f73f113b7424b15 5.9s
=> => extracting sha256:cf450bbc2f912c0a2236789627065e3e96b53d51763e7182653557d43ba98ba7 0.0s
=> => extracting sha256:e806dd517e77346add853708010a931c5d5b220c30f9b84125e253e69858be34 0.0s
=> => extracting sha256:a9eb4219fce39e31343dd629ad7697f6a46e41e25ff4933709d7bf5e41bf685 2.3s
=> => extracting sha256:9cc72bd2d4f50a0e775691c7198a1af3919a9b96f39b86cb44bd8aa5f946d 15.7s
=> => extracting sha256:6a0550e4123ec5614f97f69abb5ca767c1c43b213a52af3087ab15bba018ae9c 0.0s
=> => extracting sha256:3a6e466541337435bc0d7230b38695810ab694d60dcca510462a5e187e5d52c 0.0s
=> [2/3] RUN pip install scikit-learn numpy pandas fpdf
=> [3/3] WORKDIR /usr/app/src
=> => exporting to image
=> => writing image sha256:edd5077a5273c51a63bd28aa0889bea1fe3d6cd0f27081907b09f83e3c4a8cf6
=> => naming to docker.io/library/our_tensorflow_container
What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\lyuba\data_engineering\sde\docker>
```

Рисунок 8 – Сборка образа

После успешной сборки, необходимо произвести проверку того, что 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 our\_tensorflow\_container won2013/our\_tensorflow\_container:1.0

```
PS C:\Users\lyuba\data_engineering\sde\docker> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
our_tensorflow_container   latest             edd5877e5273       5 minutes ago      2.28GB
mlflow-web             latest             58618f29cfae       5 days ago         765MB
airflow-airflow-worker   latest             9afe4efe216c       5 days ago         1.86GB
airflow-airflow-scheduler latest             eae7c388cb57       5 days ago         1.86GB
airflow-airflow-init     latest             9dc0b15ec77b       5 days ago         1.86GB
airflow-airflow-webserver latest             178cd5f0e23e       5 days ago         1.86GB
airflow-airflow-triggerer latest             4088dffb197b       5 days ago         1.86GB
postgres               15                ad7de17c154c       6 days ago         426MB
postgres               13                70ff79bd5c49       6 days ago         419MB
minio/mc                latest            cb79163bd419       10 days ago        81.4MB
minio/minio             latest            afadd2395ca6       2 weeks ago        165MB
proxy-system-5-file-server-2 latest            b6885acff985       4 weeks ago        994MB
proxy-system-5-file-server-1 latest            b97e2311ceb2       4 weeks ago        994MB
compose-4-file-server   latest            0beccbd2efed       4 weeks ago        994MB
python-server           0.1              814589b76438       4 weeks ago        994MB
python-mount            0.1              9b84734c645f       4 weeks ago        994MB
simple-python            0.1              f35d162fd549       4 weeks ago        125MB
devops_labs             latest            2b525fef2842       4 weeks ago        1.11GB
won2013/devops_labs     latest            2b525fef2842       4 weeks ago        1.11GB
redis                   latest            6c199afc1dae       7 weeks ago        117MB
alpine                  latest            63b790fccc90       2 months ago       7.8MB
gcr.io/k8s-minikube/kicbase v0.0.45          ae0d8e1d4642       2 months ago       1.28GB
docker                  24-dind          e31dbb0fb5be       5 months ago       350MB
apache/nifi             1.23.2           81455911cd05       15 months ago      1.94GB
dpape/pgadmin4          7.6              881febbc9e93       15 months ago      534MB
nshou/elasticsearch-kibana kibana7          17f031ca3406       19 months ago      1.18GB
docker                  19.03.12-dind   66dc2d45749a       4 years ago        226MB
nginx                   1.17             9beeba249f3e       4 years ago        127MB
mysql/mysql-server      5.7.28           c8c8ef4f3c81       5 years ago        310MB
PS C:\Users\lyuba\data_engineering\sde\docker> docker tag our_tensorflow_container won2013/our_tensorflow_container:1.0
PS C:\Users\lyuba\data_engineering\sde\docker>
```

Рисунок 9 – Присвоение тега образу

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

`$ docker push won2013/our_tensorflow_container:1.0`

```
Windows PowerShell
Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
PS C:\Users\lyuba\data_engineering\sde\docker> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
our_tensorflow_container   latest             edd5877e5273       5 minutes ago      2.28GB
mlflow-web             latest             58618f29cfae       5 days ago         765MB
airflow-airflow-worker   latest             9afe4efe216c       5 days ago         1.86GB
airflow-airflow-scheduler latest             eae7c388cb57       5 days ago         1.86GB
airflow-airflow-init     latest             9dc0b15ec77b       5 days ago         1.86GB
airflow-airflow-webserver latest             178cd5f0e23e       5 days ago         1.86GB
airflow-airflow-triggerer latest             4088dffb197b       5 days ago         1.86GB
postgres               15                ad7de17c154c       6 days ago         426MB
postgres               13                70ff79bd5c49       6 days ago         419MB
minio/mc                latest            cb79163bd419       10 days ago        81.4MB
minio/minio             latest            afadd2395ca6       2 weeks ago        165MB
proxy-system-5-file-server-2 latest            b6885acff985       4 weeks ago        994MB
proxy-system-5-file-server-1 latest            b97e2311ceb2       4 weeks ago        994MB
compose-4-file-server   latest            0beccbd2efed       4 weeks ago        994MB
python-server           0.1              814589b76438       4 weeks ago        994MB
python-mount            0.1              9b84734c645f       4 weeks ago        994MB
simple-python            0.1              f35d162fd549       4 weeks ago        125MB
devops_labs             latest            2b525fef2842       4 weeks ago        1.11GB
won2013/devops_labs     latest            2b525fef2842       4 weeks ago        1.11GB
redis                   latest            6c199afc1dae       7 weeks ago        117MB
alpine                  latest            63b790fccc90       2 months ago       7.8MB
gcr.io/k8s-minikube/kicbase v0.0.45          ae0d8e1d4642       2 months ago       1.28GB
docker                  24-dind          e31dbb0fb5be       5 months ago       350MB
apache/nifi             1.23.2           81455911cd05       15 months ago      1.94GB
dpape/pgadmin4          7.6              881febbc9e93       15 months ago      534MB
nshou/elasticsearch-kibana kibana7          17f031ca3406       19 months ago      1.18GB
docker                  19.03.12-dind   66dc2d45749a       4 years ago        226MB
nginx                   1.17             9beeba249f3e       4 years ago        127MB
mysql/mysql-server      5.7.28           c8c8ef4f3c81       5 years ago        310MB
PS C:\Users\lyuba\data_engineering\sde\docker> docker tag our_tensorflow_container won2013/our_tensorflow_container:1.0
PS C:\Users\lyuba\data_engineering\sde\docker> docker push won2013/our_tensorflow_container:1.0
The push refers to repository [docker.io/won2013/our_tensorflow_container]
776a17eaae4f: Pushed
39e1244f1d33: Pushed
1d5a0ab2eb8d: Mounted from tensorflow/tensorflow
2d72a8dc5a00: Mounted from tensorflow/tensorflow
ba084a206cde: Mounted from tensorflow/tensorflow
a27cc52296f2: Mounted from tensorflow/tensorflow
7903faa1a58: Mounted from tensorflow/tensorflow
aa0b675d745e: Mounted from tensorflow/tensorflow
fd51330ca3a3: Mounted from tensorflow/tensorflow
882a1c3aed8: Mounted from tensorflow/tensorflow
b279b22d17a6: Mounted from tensorflow/tensorflow
d749eff901f9: Mounted from tensorflow/tensorflow
89353afe17dd: Mounted from tensorflow/tensorflow
2573e0d81582: Mounted from tensorflow/tensorflow
1.0: digest: sha256:83766420c5229341944baf1f3769adba29539c43db5e43ae0e079b31b4919c64 size: 3250
PS C:\Users\lyuba\data_engineering\sde\docker>
```

Рисунок 10 – Отправка образа в DockerHub

#### Шаг 4. Подготовка DAG-а.

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

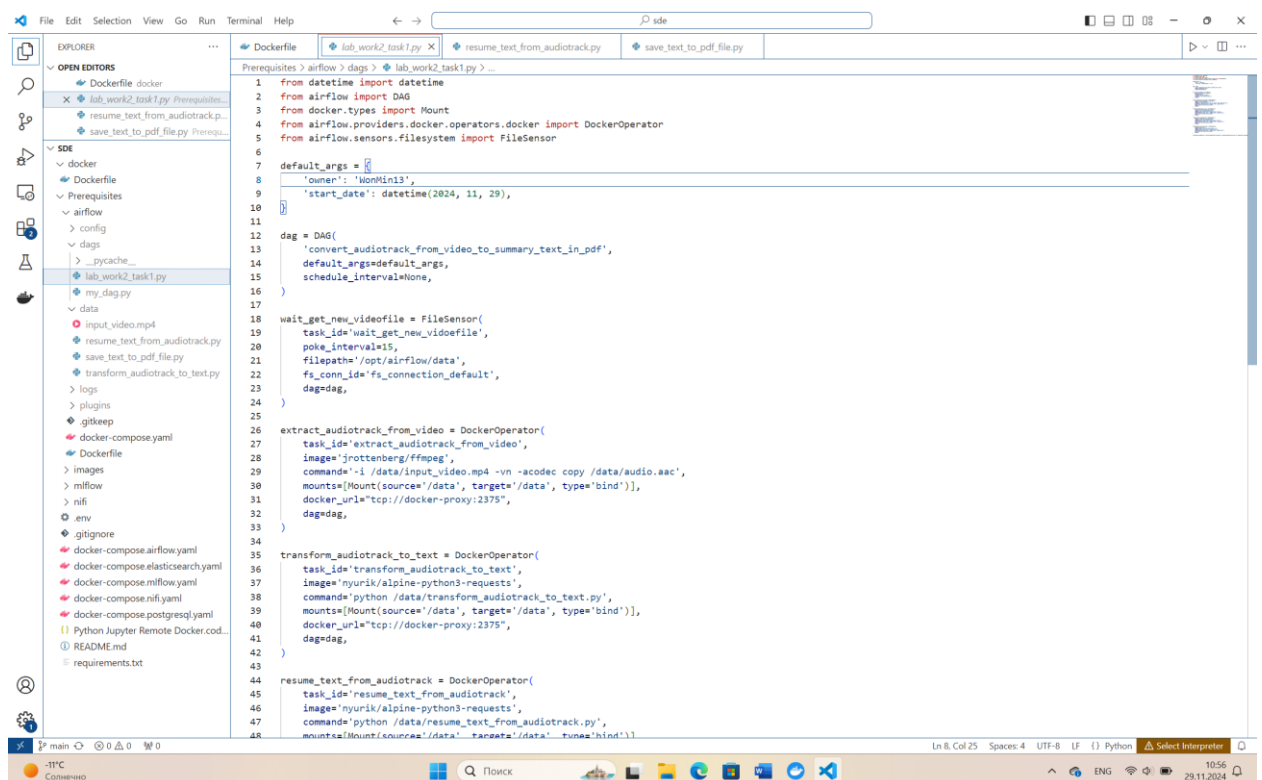
*wait\_get\_new\_videofile* – осуществляет «прослушивание» указанной директории, на предмет появления в ней видеофайла, который будет принят далее в работу.

*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.



```
1 from datetime import datetime
2 from airflow import DAG
3 from docker.types import Mount
4 from airflow.providers.docker.operators.docker import DockerOperator
5 from airflow.sensors.filesystem import FileSensor
6
7 default_args = {
8     'owner': 'MonMin13',
9     'start_date': datetime(2024, 11, 29),
10 }
11
12 dag = DAG(
13     'convert_audiotrack_from_video_to_summary_text_in_pdf',
14     default_args=default_args,
15     schedule_interval=None,
16 )
17
18 wait_get_new_videofile = FileSensor(
19     task_id='wait_get_new_videofile',
20     poke_interval=15,
21     filepath='/opt/airflow/data',
22     fs_conn_id='fs_connection_default',
23     dag=dag,
24 )
25
26 extract_audiotrack_from_video = DockerOperator(
27     task_id='extract_audiotrack_from_video',
28     image='jrottenberg/ffmpeg',
29     command='-i /data/input_video.mp4 -vn -acodec copy /data/audio.aac',
30     mounts=[Mount(source='/data', target='/data', type='bind')],
31     docker_url='tcp://docker-proxy:2375',
32     dag=dag,
33 )
34
35 transform_audiotrack_to_text = DockerOperator(
36     task_id='transform_audiotrack_to_text',
37     image='nyurik/alpine-python3-requests',
38     command='python /data/transform_audiotrack_to_text.py',
39     mounts=[Mount(source='/data', target='/data', type='bind')],
40     docker_url='tcp://docker-proxy:2375',
41     dag=dag,
42 )
43
44 resume_text_from_audiotrack = DockerOperator(
45     task_id='resume_text_from_audiotrack',
46     image='nyurik/alpine-python3-requests',
47     command='python /data/resume_text_from_audiotrack.py',
48     mounts=[Mount(source='/data', target='/data', type='bind')],
49     docker_url='tcp://docker-proxy:2375',
50     dag=dag,
51 )
52
53 save_get_text_from_txt_to_pdf = DockerOperator(
54     task_id='save_get_text_from_txt_to_pdf',
55     image='nyurik/alpine-python3-requests',
56     command='python /data/save_get_text_from_txt_to_pdf.py',
57     mounts=[Mount(source='/data', target='/data', type='bind')],
58     docker_url='tcp://docker-proxy:2375',
59     dag=dag,
60 )
61
62 wait_get_new_videofile.set_downstream(extract_audiotrack_from_video)
63 extract_audiotrack_from_video.set_downstream(transform_audiotrack_to_text)
64 transform_audiotrack_to_text.set_downstream(resume_text_from_audiotrack)
65 resume_text_from_audiotrack.set_downstream(save_get_text_from_txt_to_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:

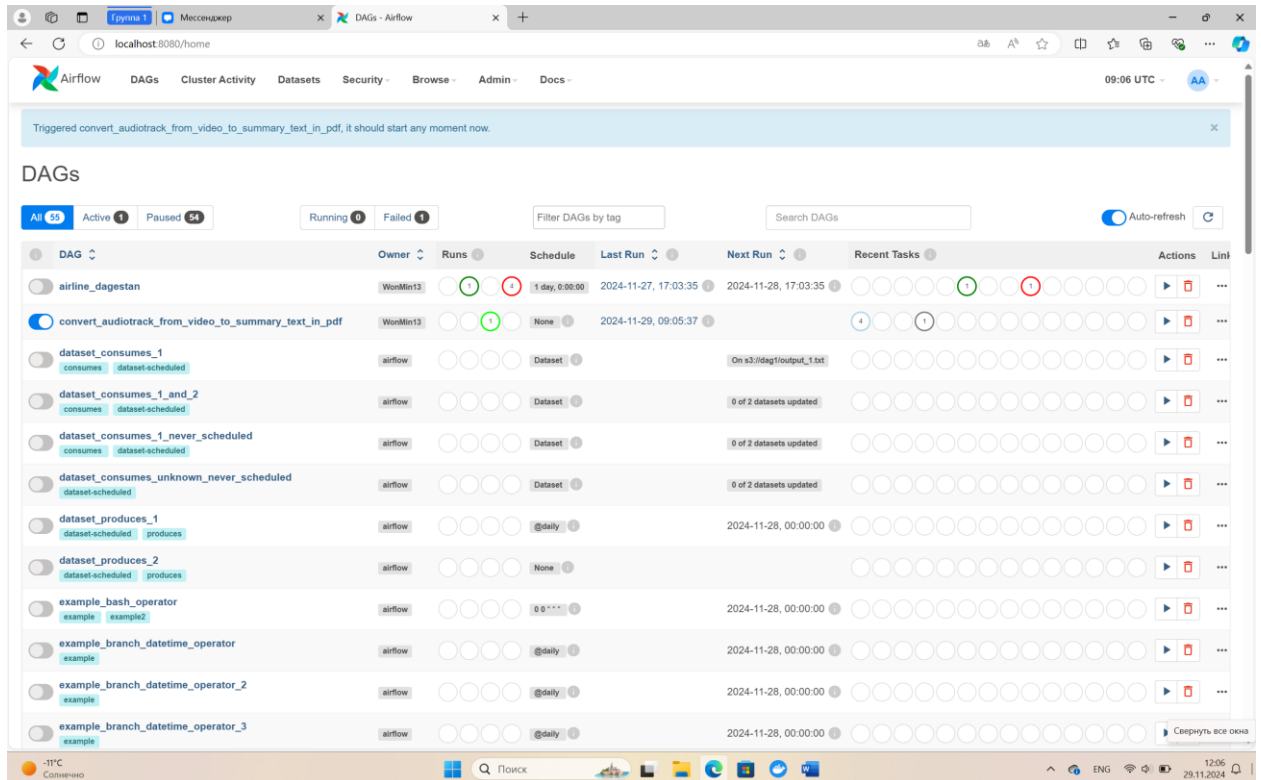
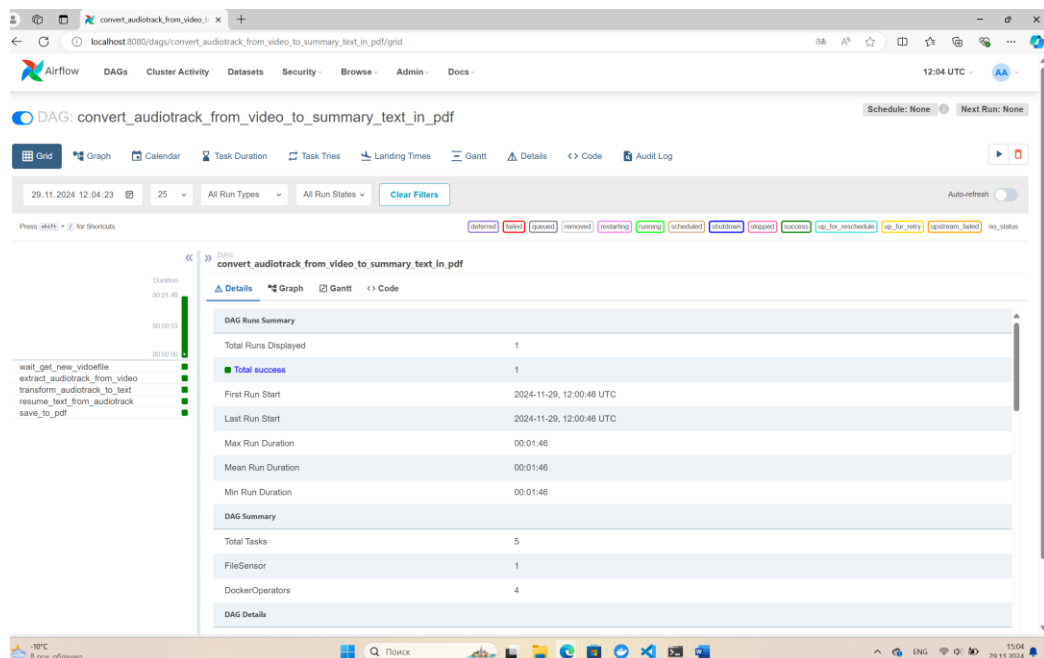


Рисунок 5 – Поиск DAG-а.

Далее запускаем наш DAG и наслаждаемся процессом.



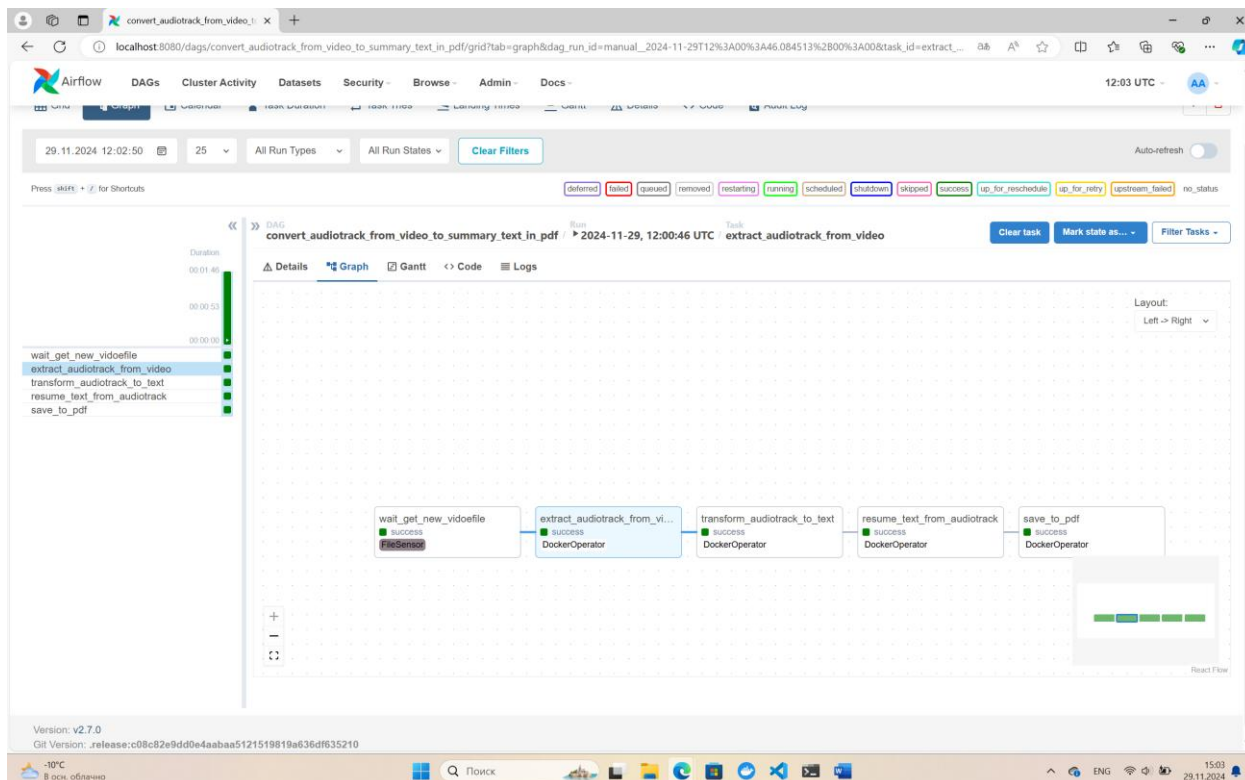


Рисунок 6 – Запуск DAG-а.

В качестве исходного видео использовался фрагмент из сериала «Гордость и предубеждение» длительностью 30 секунд. После чего мы получили аудиодорожку, которая использовалась в качестве основы для получения текстового файла.

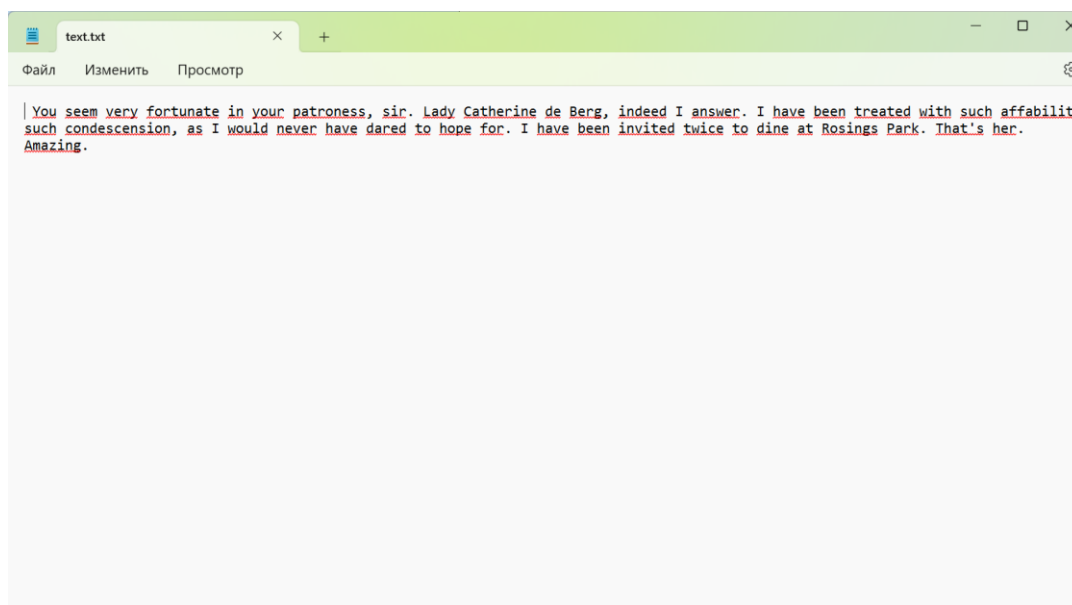


Рисунок 11 – Результат работы huggingface по преобразованию аудио в текст

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

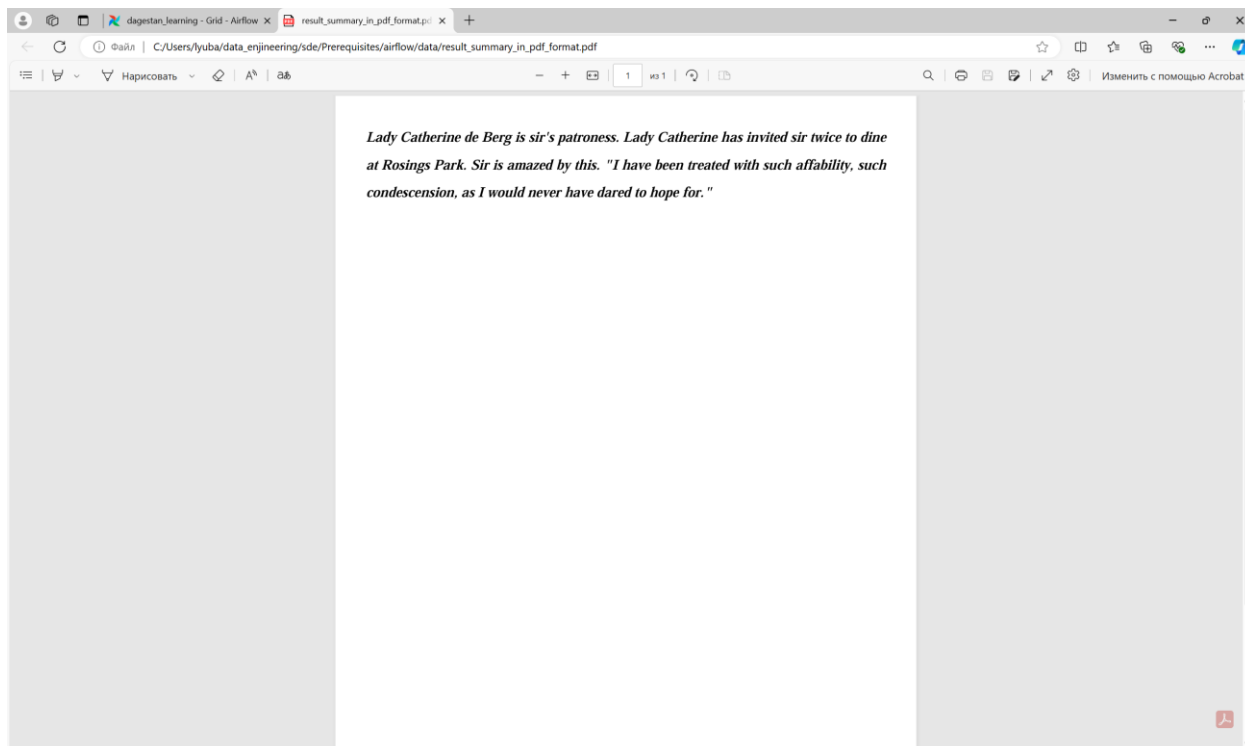


Рисунок 12 – Конспект текстового файла.

Получилось неплохо. Переходим 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

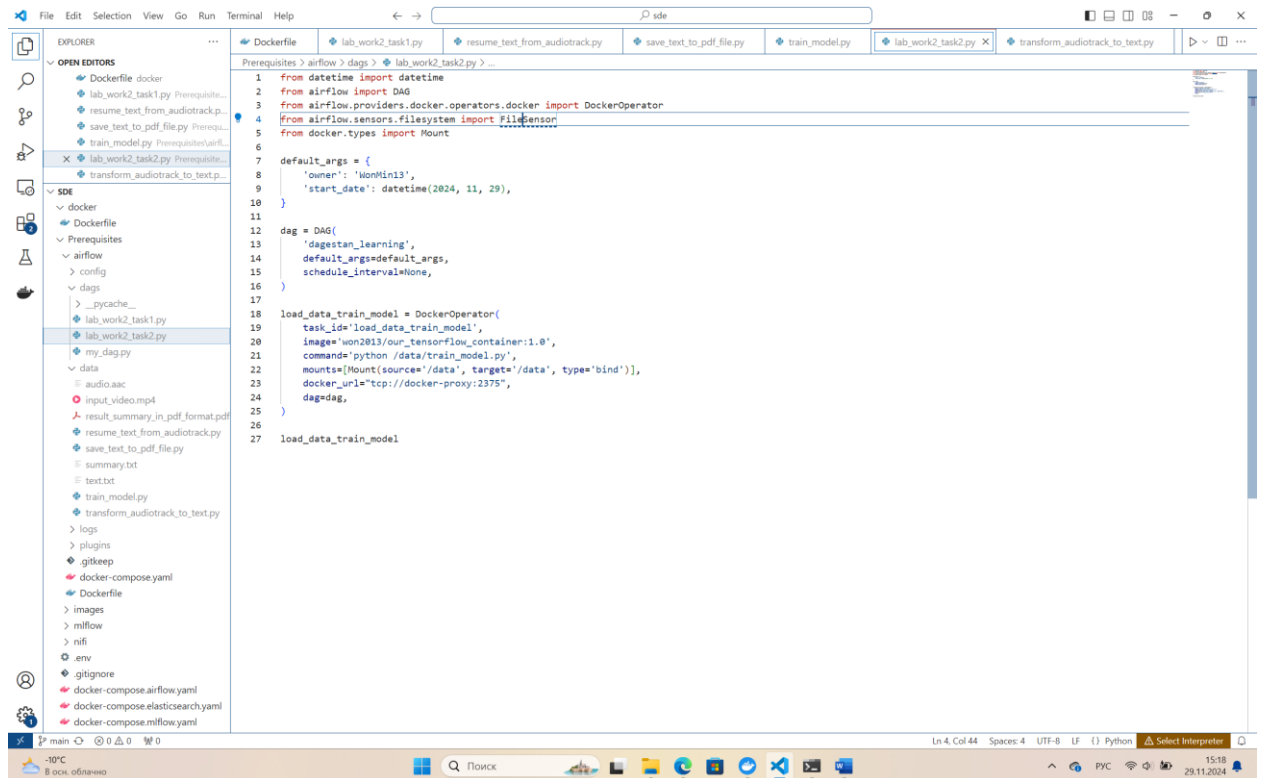


Рисунок 13 – Пайплайн

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.



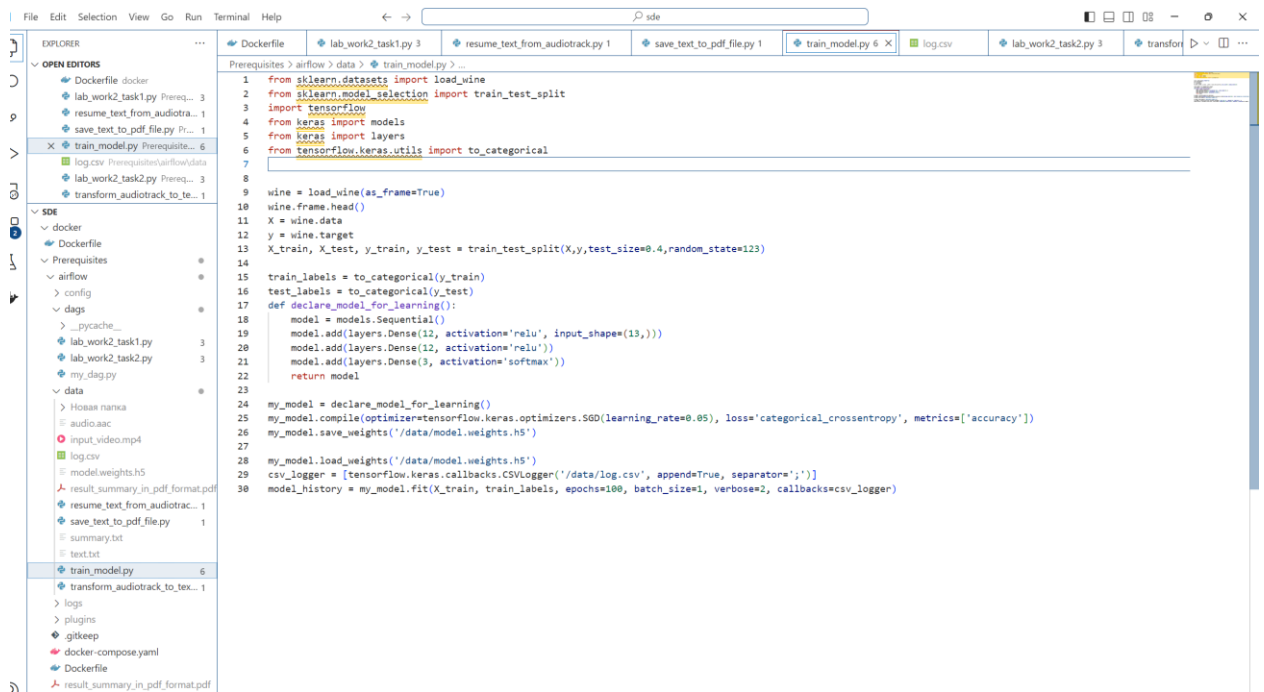
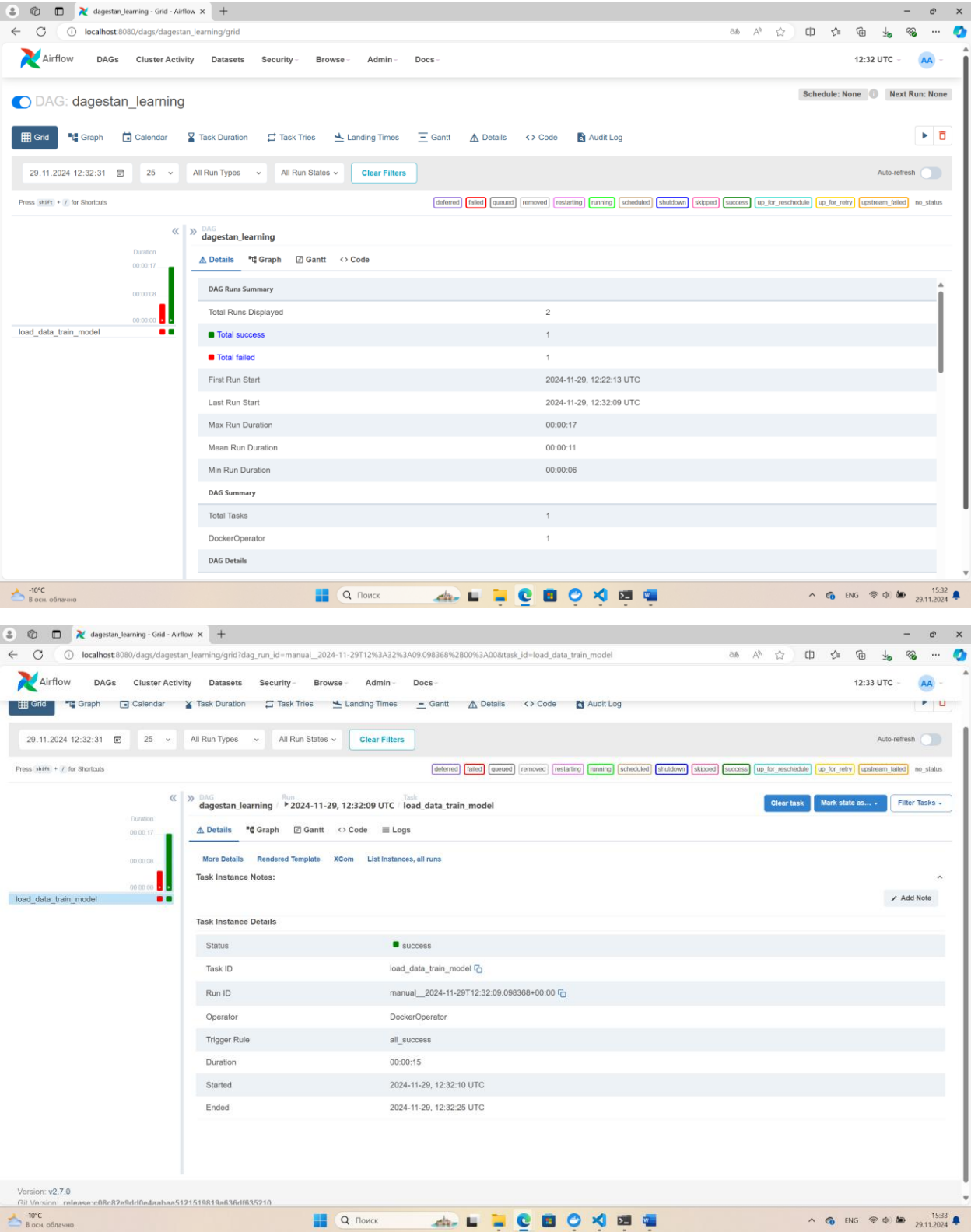


Рисунок 14 – Код обучения модели.

Шаг 2. Запуска DAG-а.

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



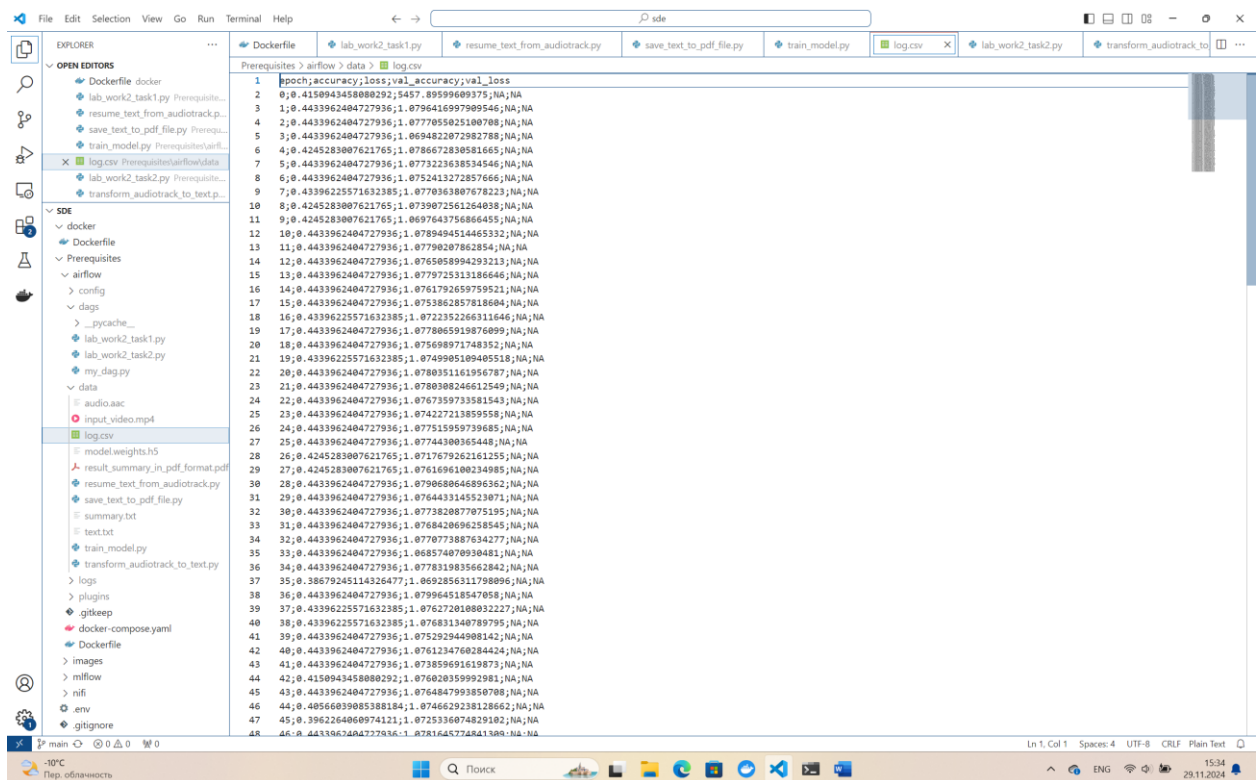


Рисунок 15 – Лог процесса обучения нейросети.

## **Заключение**

В ходе выполнения лабораторной работы получены навыки:

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