

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: Иванов И.А.

Группа: 6231-010402D

Самара 2024

## 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 необходимо в Airflow пройти по пути Admin>>Connections, как на рисунке ниже.

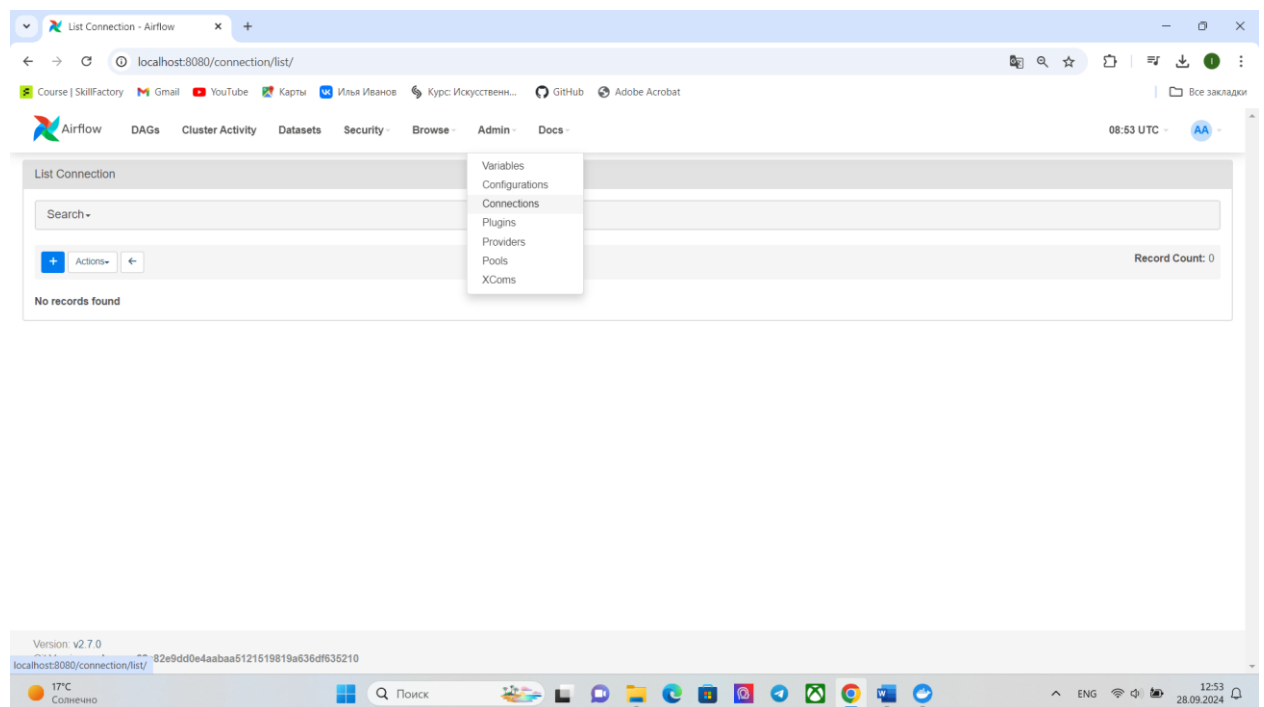


Рисунок 1 – Создание Connection

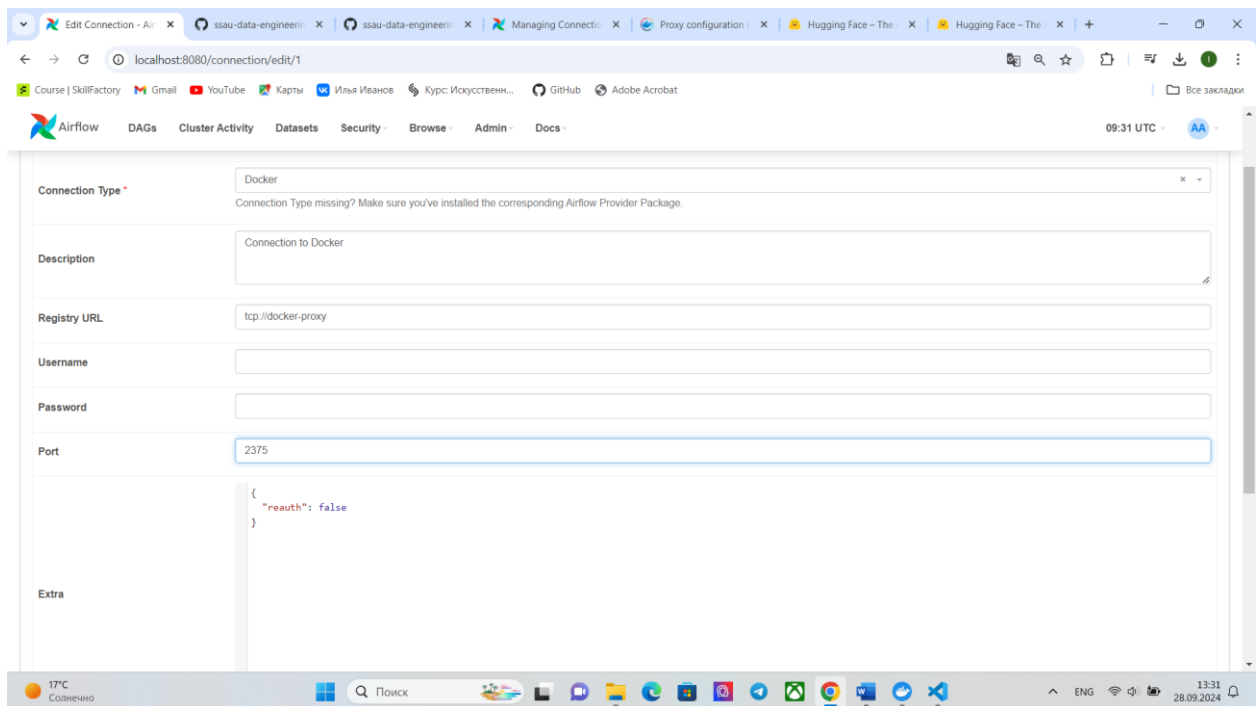


Рисунок 2 – Параметры Connection

## 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.

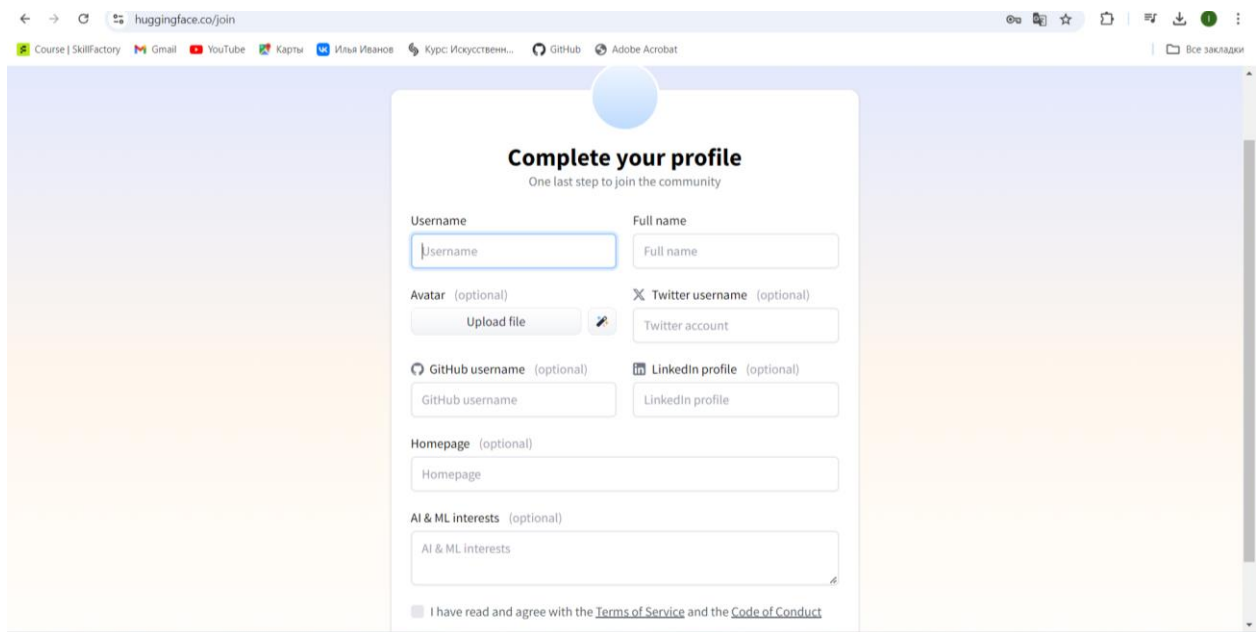


Рисунок 3 – Регистрация на huggingface

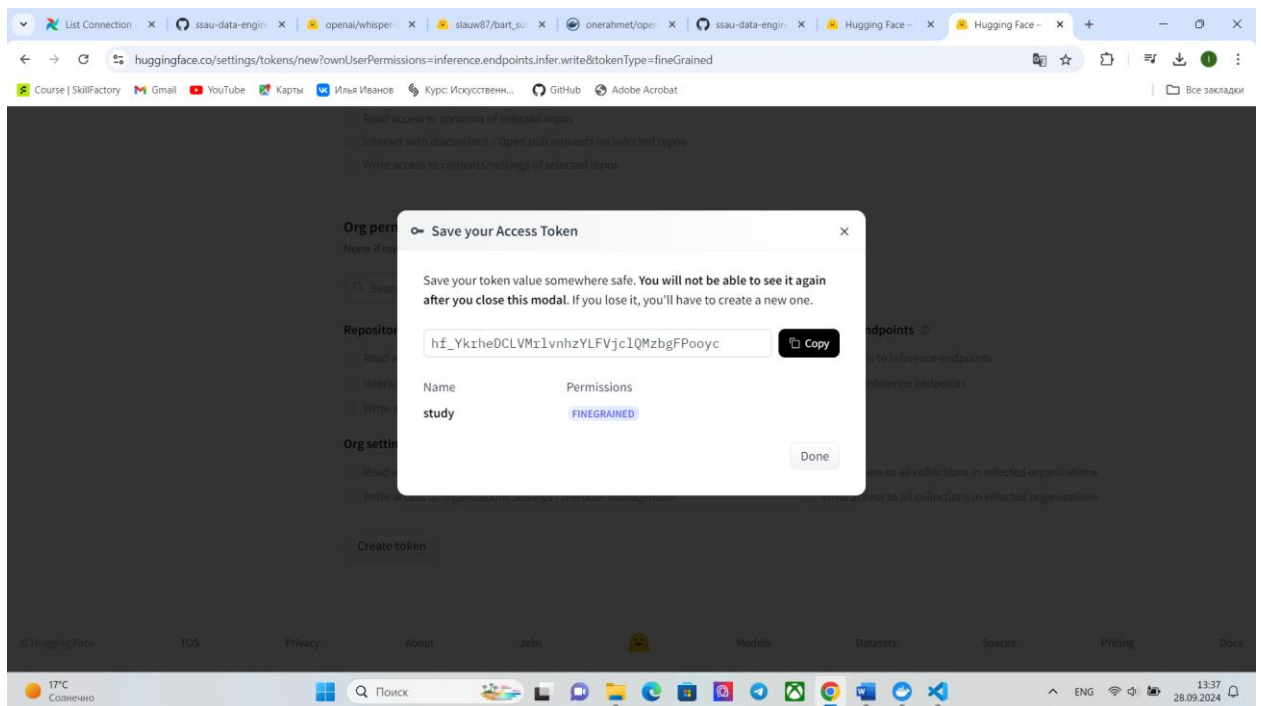


Рисунок 4 – Получение токена API

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 использовал 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.

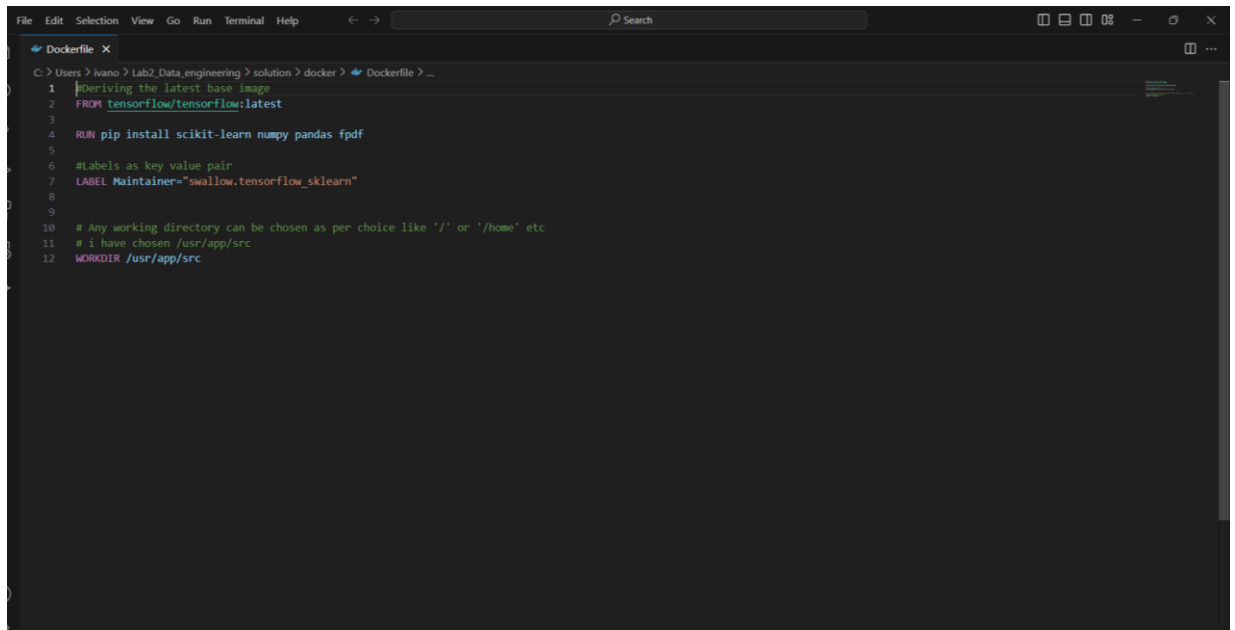


Рисунок 7 – Создание Dockerfile

Первым делом произведем сборку образа, при помощи команды:

`$ docker build -t my-app .`

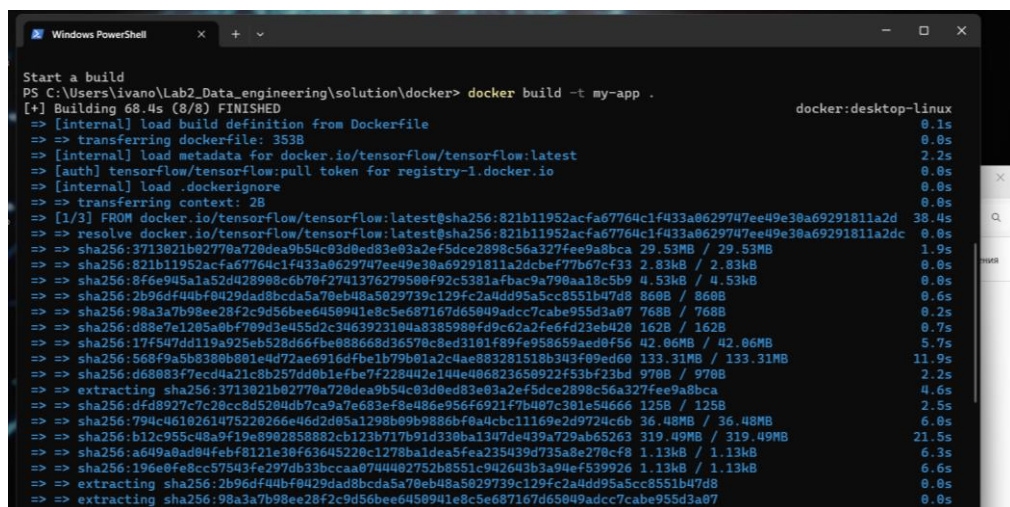


Рисунок 8 – Сборка образа

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 my-app swallow/my-app:1.0`

```
denied: requested access to the resource is denied
PS C:\Users\ivano\Lab2_Data_engineering\solution\docker> docker tag -t my-app ilyaswallow/ my-app:1.0
"docker tag" requires exactly 2 arguments.
See 'docker tag --help'.

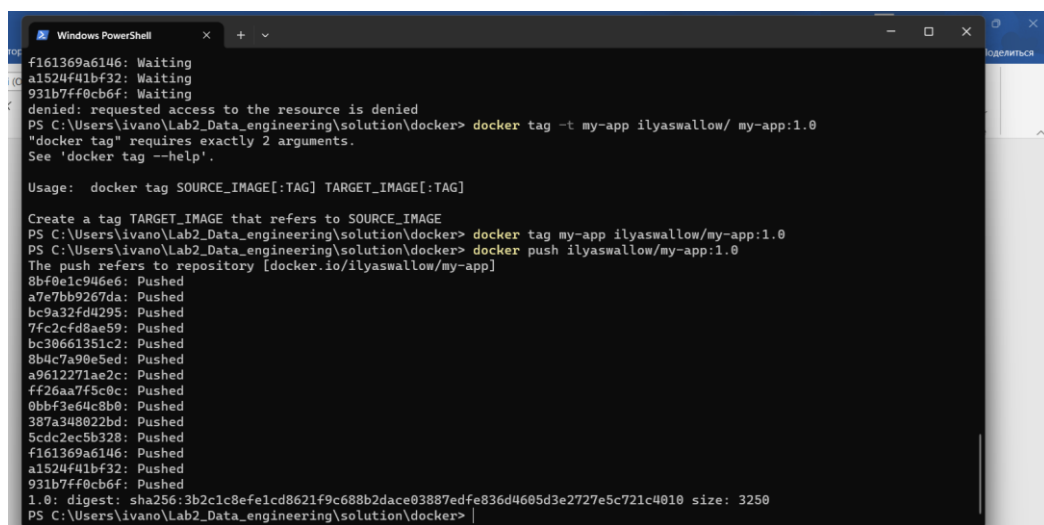
Usage: docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]

Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
PS C:\Users\ivano\Lab2_Data_engineering\solution\docker> docker tag my-app ilyaswallow/my-app:1.0
```

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
f161369a6146: Waiting
a1524f41bf32: Waiting
931b7ff0cb6f: Waiting
denied: requested access to the resource is denied
PS C:\Users\ivano\Lab2_Data_engineering\solution\docker> docker tag -t my-app ilyaswallow/ my-app:1.0
"docker tag" requires exactly 2 arguments.
See 'docker tag --help'.

Usage: docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]

Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
PS C:\Users\ivano\Lab2_Data_engineering\solution\docker> docker tag my-app ilyaswallow/my-app:1.0
PS C:\Users\ivano\Lab2_Data_engineering\solution\docker> docker push ilyaswallow/my-app:1.0
The push refers to repository [docker.io/ilyaswallow/my-app]
8bf0e1e9d6e6: Pushed
a7e7bb9267da: Pushed
bc9a32fd4205: Pushed
7fc2cf48ae59: Pushed
bc30661351c2: Pushed
8b4c7a90e5ed: Pushed
a9612271ae2c: Pushed
ff26aa7f5c0c: Pushed
0bbf3e64c8b0: Pushed
387a348022bd: Pushed
5cdc2ec5b328: Pushed
f161369a6146: Pushed
a1524f41bf32: Pushed
931b7ff0cb6f: Pushed
1.0: digest: sha256:3b2c1c8efelcd8621f9c688b2dace03887edfe836d4605d3e2727e5c721c4010 size: 3250
PS C:\Users\ivano\Lab2_Data_engineering\solution\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-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:

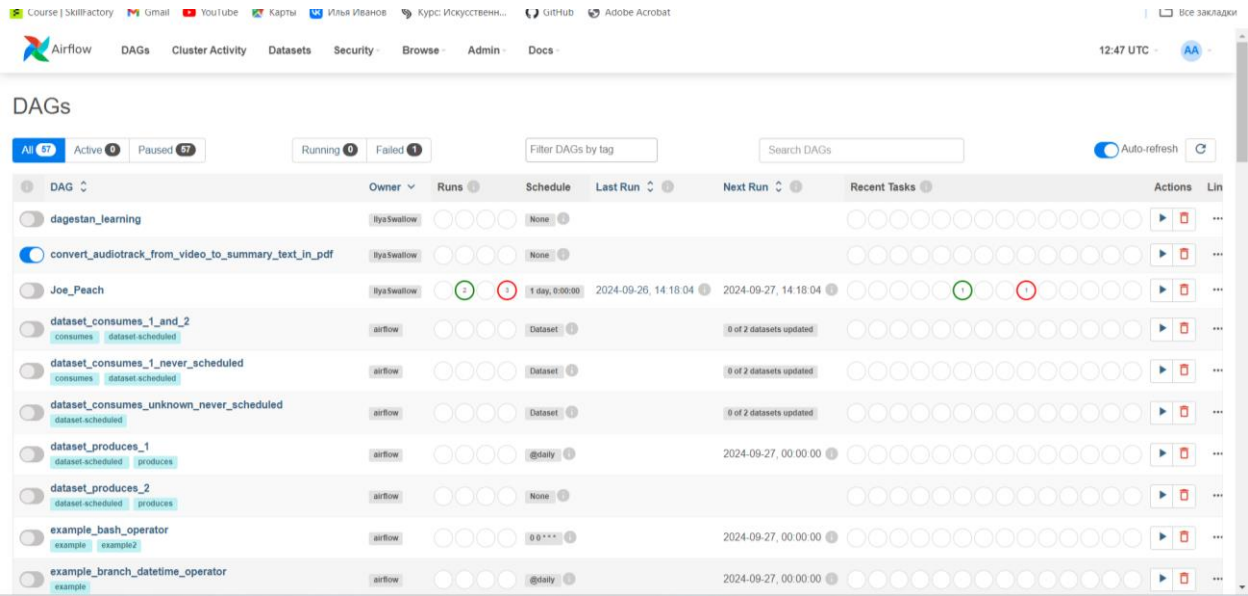


Рисунок 5 – Поиск DAG-а.

Далее запускаем наш DAG.



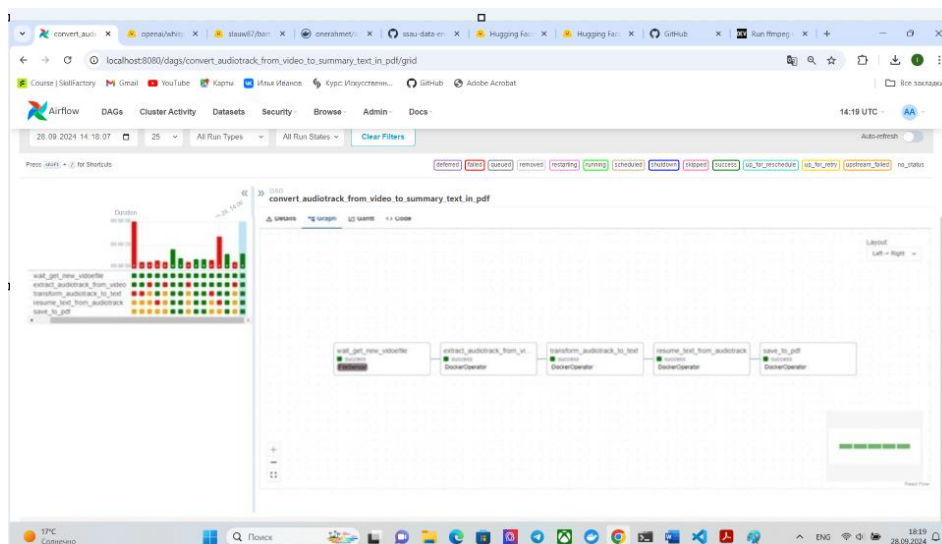


Рисунок 6 – Запуск DAG-а.

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.

---

*Rony Child's daughter was ruined by her father. The father came to the Council of Help and asked him to pay money for the crime. Rony Child refused to do it because he didn't want to get into trouble with his daughter's father. He will pay the father whatever he owes him.*

---

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

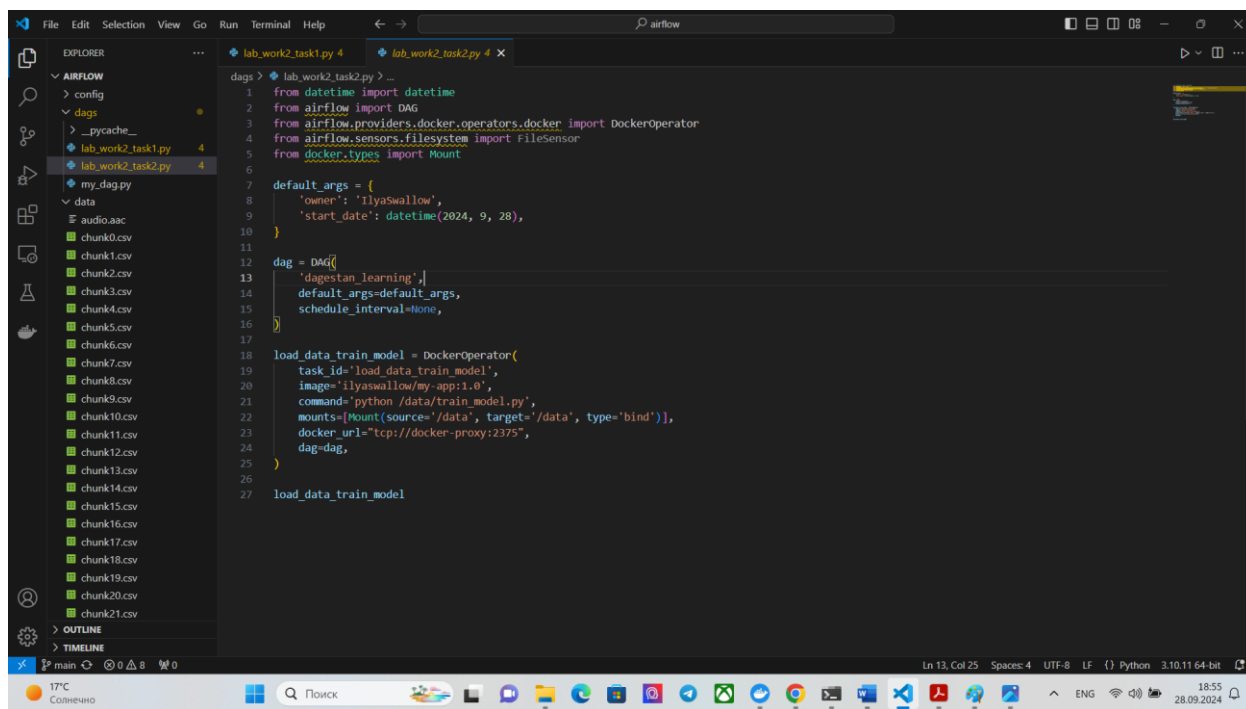
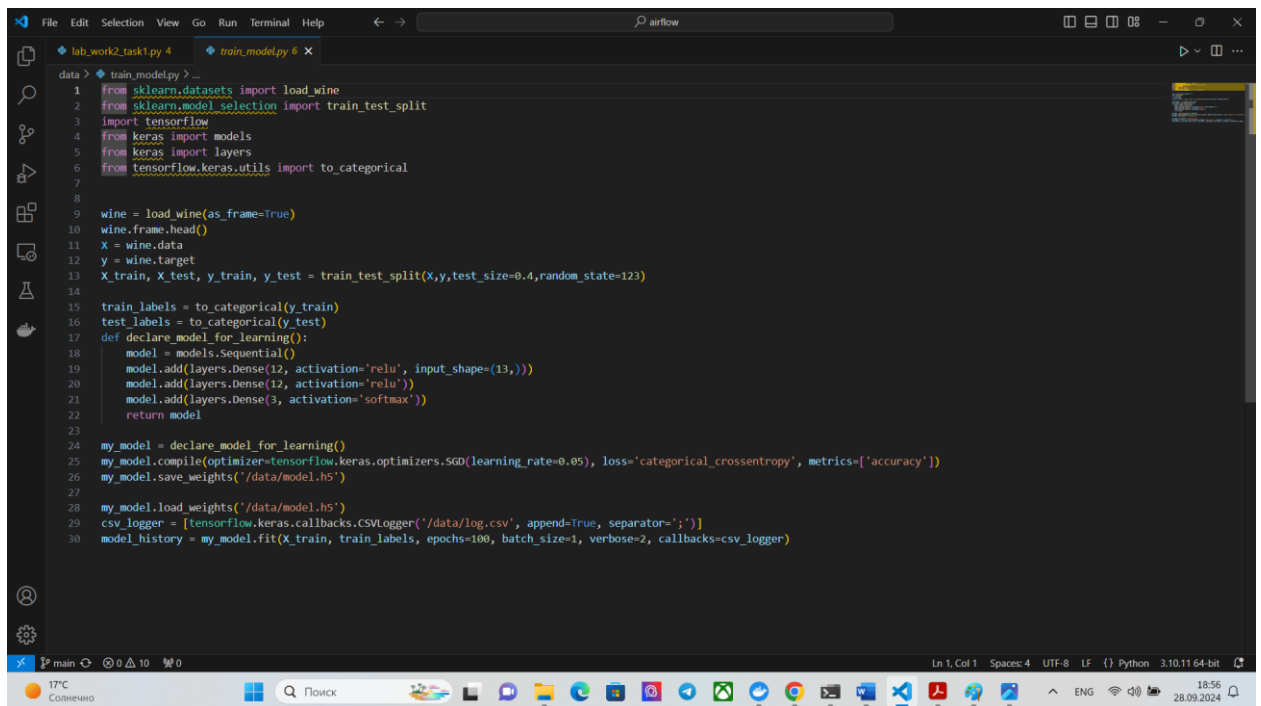


Рисунок 13 - Пайплайн

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.

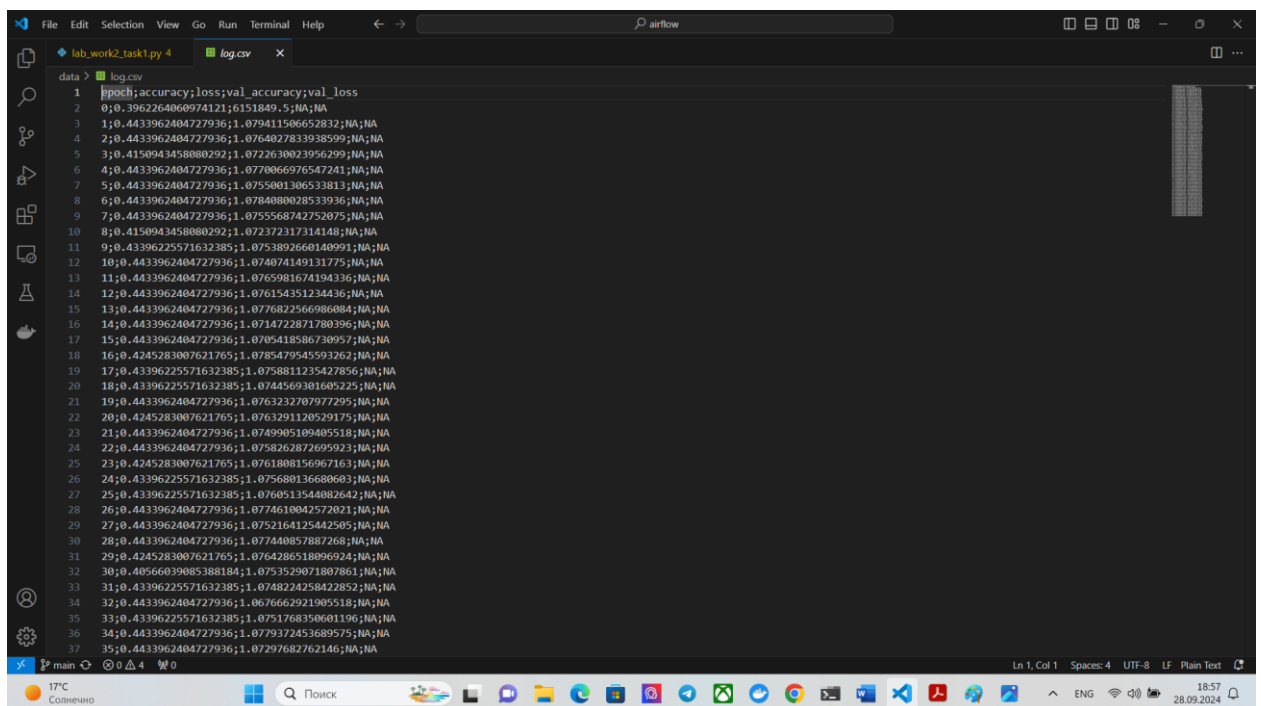


```
1 from sklearn.datasets import load_wine
2 from sklearn.model_selection import train_test_split
3 import tensorflow
4 from keras import models
5 from keras import layers
6 from tensorflow.keras.utils import to_categorical
7
8 wine = load_wine(as_frame=True)
9 wine.head()
10 X = wine.data
11 y = wine.target
12 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=123)
13
14 train_labels = to_categorical(y_train)
15 test_labels = to_categorical(y_test)
16
17 def declare_model_for_learning():
18     model = models.Sequential()
19     model.add(layers.Dense(12, activation='relu', input_shape=(13,)))
20     model.add(layers.Dense(12, activation='relu'))
21     model.add(layers.Dense(3, activation='softmax'))
22     return model
23
24 my_model = declare_model_for_learning()
25 my_model.compile(optimizer='tensorflow.keras.optimizers.SGD(learning_rate=0.05)', loss='categorical_crossentropy', metrics=['accuracy'])
26 my_model.save_weights('/data/model.h5')
27
28 my_model.load_weights('/data/model.h5')
29 csv_logger = [tensorflow.keras.callbacks.CSVLogger('/data/log.csv', append=True, separator=';')]
30 model_history = my_model.fit(X_train, train_labels, epochs=100, batch_size=1, verbose=2, callbacks=csv_logger)
```

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:



```
1 epoch;accuracy;loss;val_accuracy;val_loss
2 0;0.396226406974121;6151849.5;NA;NA
3 1;0.4433962404727936;1.079411506652832;NA;NA
4 2;0.4433962404727936;1.0764027833938959;NA;NA
5 3;0.4150943458080292;1.0722630023956299;NA;NA
6 4;0.4433962404727936;1.0770066976547241;NA;NA
7 5;0.4433962404727936;1.0755001306533813;NA;NA
8 6;0.4433962404727936;1.0784080028533936;NA;NA
9 7;0.4433962404727936;1.0755568742752075;NA;NA
10 8;0.4150943458080292;1.07232317314148;NA;NA
11 9;0.43396225571632385;1.0753892660140991;NA;NA
12 10;0.4433962404727936;1.074874149131775;NA;NA
13 11;0.4433962404727936;1.0765981674104336;NA;NA
14 12;0.4433962404727936;1.076154351234436;NA;NA
15 13;0.4433962404727936;1.0776822566086804;NA;NA
16 14;0.4433962404727936;1.0714722871780396;NA;NA
17 15;0.4433962404727936;1.0705418586730957;NA;NA
18 16;0.4245283007621765;1.0785479545593262;NA;NA
19 17;0.443396225571632385;1.0758811235427856;NA;NA
20 18;0.443396225571632385;1.0744569301605225;NA;NA
21 19;0.4433962404727936;1.0763232707977295;NA;NA
22 20;0.4245283007621765;1.0763291120529175;NA;NA
23 21;0.4433962404727936;1.0749905109405518;NA;NA
24 22;0.4433962404727936;1.0758262872695923;NA;NA
25 23;0.4245283007621765;1.0761808156967163;NA;NA
26 24;0.443396225571632385;1.075680136680603;NA;NA
27 25;0.443396225571632385;1.0760513544082642;NA;NA
28 26;0.4433962404727936;1.0774610042572021;NA;NA
29 27;0.4433962404727936;1.0752164125442505;NA;NA
30 28;0.4433962404727936;1.077440857887268;NA;NA
31 29;0.4245283007621765;1.0764286518096924;NA;NA
32 30;0.40566039085388184;1.0753529071807861;NA;NA
33 31;0.443396225571632385;1.074822425422852;NA;NA
34 32;0.4433962404727936;1.0676662921905518;NA;NA
35 33;0.443396225571632385;1.0751768350601196;NA;NA
36 34;0.4433962404727936;1.0779372453689575;NA;NA
37 35;0.4433962404727936;1.07297682762146;NA;NA
```

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. Работс сетями на huggingface