

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, математики и электроники  
Факультет информатики  
Кафедра технической кибернетики

**Отчет по лабораторной работе № 2  
по курсу «Инженерия данных»**

Тема: «Инференс и обучение НС»

Выполнил Самигуллин Равиль  
Группа 6133 – 010402D  
Преподаватель Парингер Р.А

**Самара**

## Содержание

Содержание.....	2
Техническое задание.....	3
Задание 1: Пайплайн для инференса данных. ....	5
Задание 2: Пайплайн для обучения модели.....	14
ЗАКЛЮЧЕНИЕ .....	21

## Техническое задание

### 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:

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.

### 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:

Читать набор файлов из определенного источника (файловой системы, сетевого интерфейса и т.д.).

Формировать пакет данных для обучения модели.

Обучать модель.


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

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

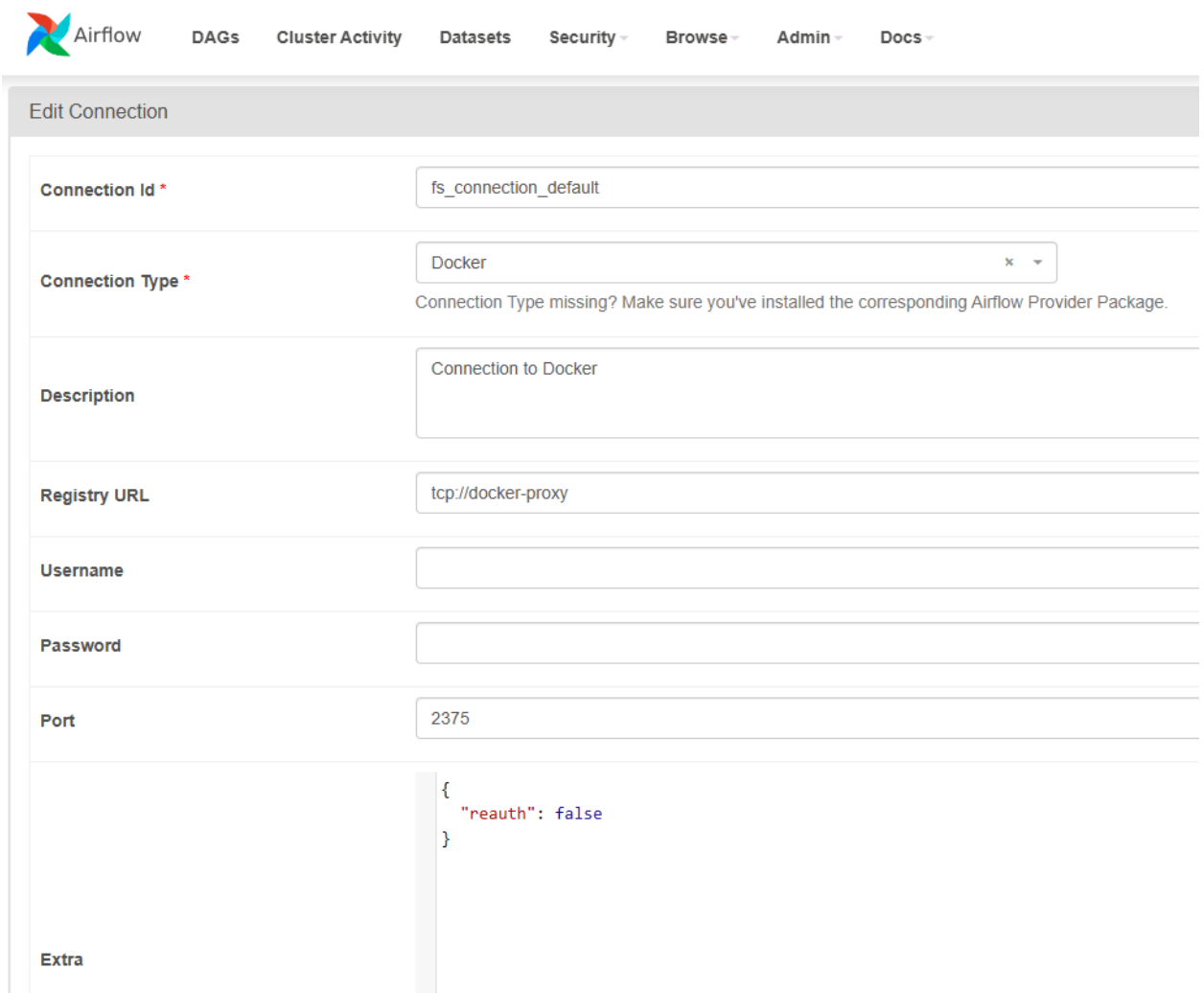
## Задание 1: Пайплайн для инференса данных.

### 1. Подключение dockeroperator для выполнения ЛР.

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.



	Conn Id	Conn Type	Description	Host	Port	Is Encrypted	Is Extra Encrypted
<input type="checkbox"/>	fs_connection_default	docker	Connection to Docker	tcp://docker-proxy	2375	False	False



**Connection Id \*** fs\_connection\_default

**Connection Type \*** Docker  
Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.

**Description** Connection to Docker

**Registry URL** tcp://docker-proxy

**Username**

**Password**

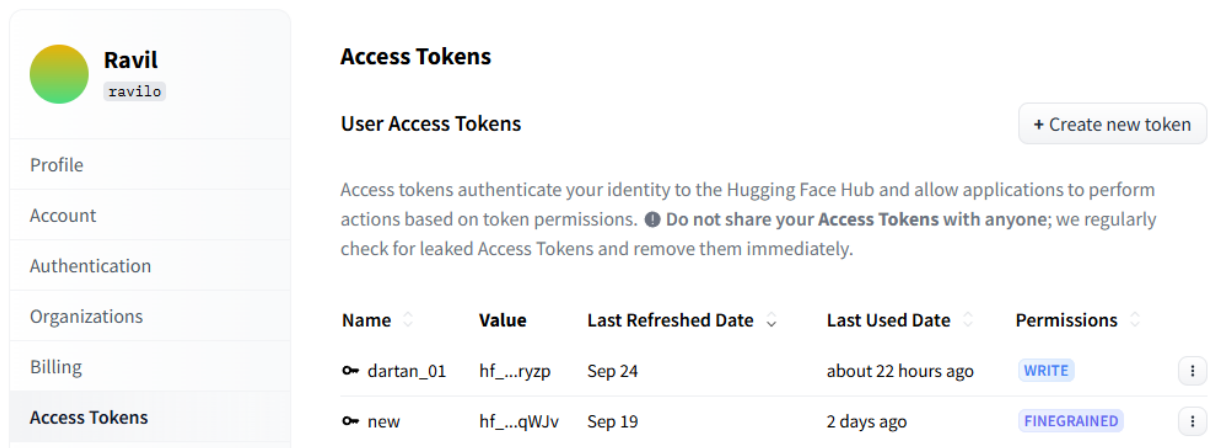
**Port** 2375

**Extra**

```
{  
  "reauth": false  
}
```

## 2. Взять api с hugging face

Я уже был зарегистрирован на hugging face, и использовал оттуда токен на



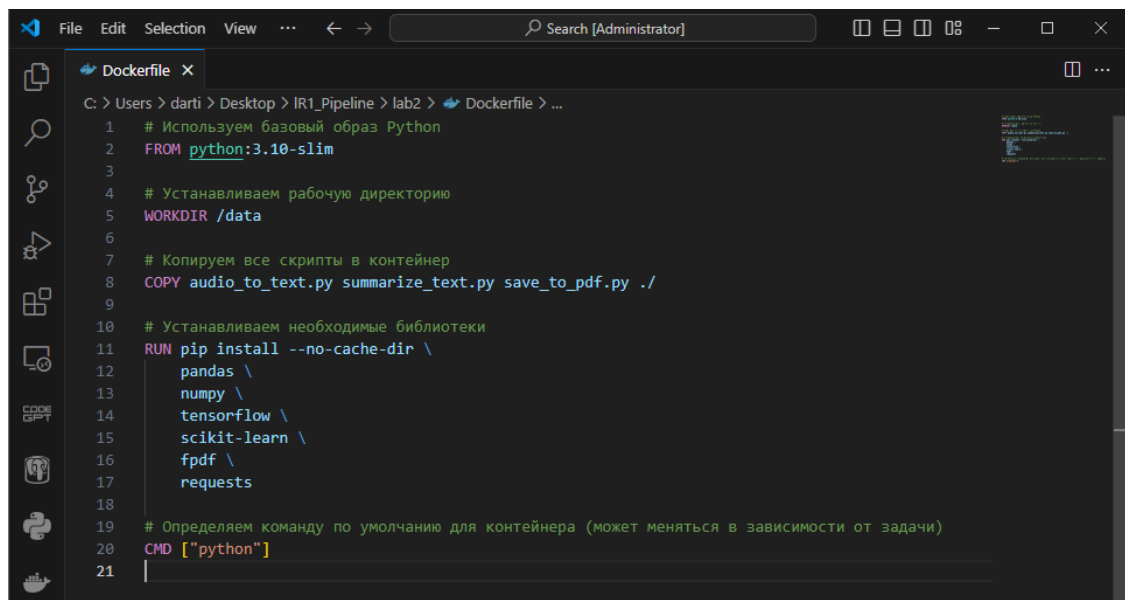
The screenshot shows the Hugging Face user interface. On the left is a sidebar with the user's profile 'Ravil' and navigation links: Profile, Account, Authentication, Organizations, Billing, and Access Tokens (which is highlighted). The main area is titled 'Access Tokens' and contains a section 'User Access Tokens' with a '+ Create new token' button. Below this is a warning: 'Access tokens authenticate your identity to the Hugging Face Hub and allow applications to perform actions based on token permissions. Do not share your Access Tokens with anyone; we regularly check for leaked Access Tokens and remove them immediately.' A table lists existing tokens:

Name	Value	Last Refreshed Date	Last Used Date	Permissions
dartan_01	hf_...ryzp	Sep 24	about 22 hours ago	WRITE
new	hf_...qWJv	Sep 19	2 days ago	FINEGRAINED

Для нашей задачи нужен токен на write, так как мы не собираемся качать модель и работать через pipeline (функция transformers hugging face), а будем отправлять json.

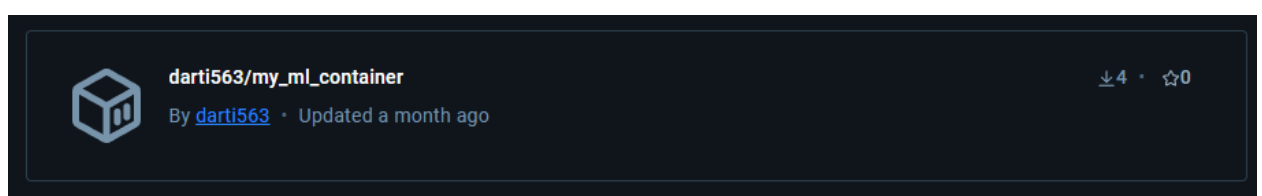
## 3. Создание docker образа

Создал докер образ, как я понял можно использовать сам контейнер или его образ из docker hub, через первый способ у меня не особо получилось поэтому я загрузил образ на докер хаб.



```
1 # Используем базовый образ Python
2 FROM python:3.10-slim
3
4 # Устанавливаем рабочую директорию
5 WORKDIR /data
6
7 # Копируем все скрипты в контейнер
8 COPY audio_to_text.py summarize_text.py save_to_pdf.py ./
9
10 # Устанавливаем необходимые библиотеки
11 RUN pip install --no-cache-dir \
12     pandas \
13     numpy \
14     tensorflow \
15     scikit-learn \
16     fpdf \
17     requests
18
19 # Определяем команду по умолчанию для контейнера (может меняться в зависимости от задачи)
20 CMD ["python"]
21
```

Докер хаб: [https://hub.docker.com/r/darti563/my\\_ml\\_container](https://hub.docker.com/r/darti563/my_ml_container)



The screenshot shows the Docker Hub page for the image 'darti563/my\_ml\_container'. It includes the Docker logo, the image name, the creator 'By darti563', and the update time 'Updated a month ago'. There are also download and star icons with counts.

В итоге копирование питон файлов в контейнер, мне не приходилось так как, я решил, что лучше монтировать папку data с скриптами, и использовать её для хранения и модификации скриптов при необходимости.

Я добавил несколько библиотек для обучения моделей в контейнер, tensorflow и requests для отправки json на сервера hugging face.

#### 4. Написание DAG.

```
dag_lr2_task_1.py > ...
1  from datetime import datetime
2  from airflow import DAG
3  from airflow.providers.docker.operators.docker import DockerOperator
4  from airflow.sensors.filesystem import FileSensor
5  from docker.types import Mount # Импортируем Mount
6
7  default_args = {
8      'owner': 'darti',
9      'start_date': datetime(2024, 1, 2),
10     'retries': 1,
11 }
12
13 dag = DAG(
14     'audio_to_pdf',
15     default_args=default_args,
16     description='DAG for extracting audio, transforming to text, summarizing, and saving as PDF',
17     schedule_interval=None,
18 )
19
20 wait_for_new_file = FileSensor(
21     task_id='wait_for_new_file',
22     poke_interval=10, # Interval to check for new files (in seconds)
23     filepath= '/opt/airflow/data',
24     fs_conn_id='fs_connection_default', # Target folder to monitor
25     dag=dag,
26 )
27
28 extract_audio = DockerOperator(
29     task_id='extract_audio',
30     image='jrottenberg/ffmpeg',
31     command='-i /data/input_video.mp4 -vn -acodec copy /data/audio.aac',
32     mounts=[Mount(source='/data', target='/data', type='bind')],
33     docker_url="tcp://docker-proxy:2375",
34     dag=dag,
35 )
36
37 transform_audio_to_text = DockerOperator(
38     task_id='transform_audio_to_text',
39     image='darti563/my_ml_container',
40     command=[
41         '/data/audio_to_text.py',
42         '/data/audio.aac',
43         '/data/text.txt'
44     ],
45     mounts=[Mount(source='/data', target='/data', type='bind')],
46     docker_url="tcp://docker-proxy:2375",
47     dag=dag,
48 )
49
```

```

summarize_text = DockerOperator(
    task_id='summarize_text',
    image='darti563/my_ml_container',
    command=[
        '/data/summarize_text.py',
        '/data/text.txt',
        '/data/summary.txt'
    ],
    mounts=[Mount(source='/data', target='/data', type='bind')],
    docker_url="tcp://docker-proxy:2375",
    dag=dag,
)

save_to_pdf = DockerOperator(
    task_id='save_to_pdf',
    image='darti563/my_ml_container',
    command=[
        '/data/save_to_pdf.py',
        '/data/summary.txt',
        '/data/result.pdf'
    ],
    mounts=[Mount(source='/data', target='/data', type='bind')],
    docker_url="tcp://docker-proxy:2375",
    dag=dag,
)

wait_for_new_file >> extract_audio >> transform_audio_to_text >> summarize_text >> save_to_pdf

```

## 5. Контейнер с ffmpeg

Я для каждой задачи кроме первой file sensor, docker operator для первой задачи было логично использовать образ ffmpeg библиотека, которая работает с аудио и видео файлами, выполняем команду

```
-i /data/input_video.mp4 -vn -acodec copy /data/audio.aac
```

## 6. Скрипты для my\_ml\_container

При этом монтируя папку /data, после для всех задач использовал ml\_container, который загрузил в хаб, и передаём в команде скрипт и переменные среды.

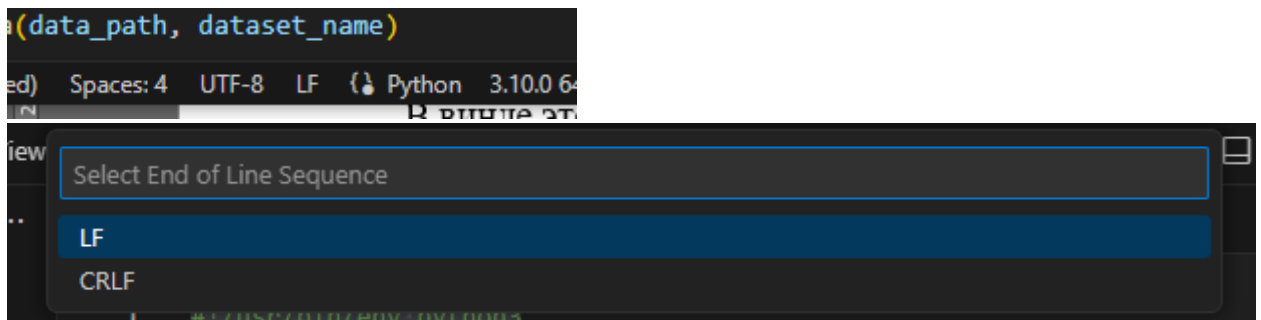
Но просто так это не заработает, в самом скрипте нужно прописать шейбенг  
#!/usr/bin/env python3

Который помогает системе выбрать сразу интерпретатор для запуска кода.

Пути к файлам мы уже используем в самом коде, тут ещё есть важный момент, так как я сижу на винде то есть проблема с end of line sequence.

В винде это CRLF (\r\n), а в системе unix которую использует контейнер это LF (\n), и при запуске скрипт может не сработать если мы не пропишем этот формат в vs code:





После того, как я всё это учел мой скрипт наконец заработал.

Теперь можно переходить к описанию самих скриптов:

- transform\_audio\_to\_text

```
#!/usr/bin/env python3
import requests
import sys
import json
💡
API_TOKEN = "REDACTED"
API_URL = "https://api-inference.huggingface.co/models/openai/whisper-small"
headers = {"Authorization": f"Bearer {API_TOKEN}"}

# Чтение входного аудиофайла, который передается как аргумент командной строки
input_file = sys.argv[1]
output_file = sys.argv[2]

# Открываем аудиофайл и отправляем его на обработку в модель Whisper
with open(input_file, "rb") as f:
    audio_data = f.read()

response = requests.post(API_URL, headers=headers, data=audio_data)

# Проверяем, успешно ли выполнен запрос
if response.status_code == 200:
    result = response.json()

    # Сохраняем результат в выходной файл
    with open(output_file, 'w') as f:
        json.dump(result, f)
else:
    print("Error:", response.text)
    sys.exit(1)
```

Тут всё просто отправляем наш аудио файл модели для анализа используя json и api token, я замазал его, а вообще обычно грузю его через dotenv.

Пути для файла и выходного файла, мы берем из команды переданной системе.

```
return self.create_container_from_config(config, name, platform)
File ~/home/airflow/.local/lib/python3.8/site-packages/docker/api/container.py, line 448, in create_container_from_config
    return self._result(res, True)
File ~/home/airflow/.local/lib/python3.8/site-packages/docker/api/client.py, line 274, in _result
    self._raise_for_status(response)
File ~/home/airflow/.local/lib/python3.8/site-packages/docker/api/client.py, line 278, in _raise_for_status
    raise create_api_error_from_http_exception(e) from e
File ~/home/airflow/.local/lib/python3.8/site-packages/docker/errors.py, line 39, in create_api_error_from_http_exception
    raise cls(e, response=response, explanation=explanation) from e
docker.errors.APIError: 400 Client Error for http://docker-proxy:2375/v1.43/containers/create: Bad Request ("invalid mount config for type "bind": bind source path does not exist: /tmp/airflowtmp1k0e692u")
During handling of the above exception, another exception occurred:
Traceback (most recent call last):
File ~/home/airflow/.local/lib/python3.8/site-packages/airflow/providers/docker/operators/docker.py, line 476, in execute
    return self._run_image()
File ~/home/airflow/.local/lib/python3.8/site-packages/airflow/providers/docker/operators/docker.py, line 347, in _run_image
    return self._run_image_with_mounts(self.mounts, add_tmp_variable=False)
File ~/home/airflow/.local/lib/python3.8/site-packages/airflow/providers/docker/operators/docker.py, line 411, in _run_image_with_mounts
    raise AirflowException("Docker container failed: {repr(result)} lines (joined_log_lines)")
airflow.exceptions.AirflowException: Docker container failed: ('Statuscode': 1) lines Error: ("error": "Model openai/whisper-small is currently loading", "estimated_time": "38.67758560180664")
[2024-09-26, 11:59:25 UTC] [taskinstance.py:1408] INFO - Marking task as UP_FOR_RETRY, dag_id=audio_to_pdf, task_id=transform_audio_to_text, execution_date=20240926T115917, start_date=20240926T115925
[2024-09-26, 11:59:25 UTC] [standard_task_runner.py:134] ERROR - Failed to execute job 177 for task transform_audio_to_text (Docker container failed: ('Statuscode': 1) lines Error: ("error": "Model openai/whisper-small is currently loading", "estimated_time": "38.67758560180664"))
[2024-09-26, 11:59:25 UTC] [local_task_runner.py:1228] INFO - Task exited with return code 1
[2024-09-26, 11:59:25 UTC] [taskinstance.py:2784] INFO - 0 downstream tasks scheduled from follow-on schedule check
```

Так же может быть так при первом запуске что модель на hugging face не доступна, это объясняется тем что сервер с моделью только запускается и был выключен так как никому не нужен

Можно поставить задержку на 60 секунд после отправки, но я решил ничего не делать так как dag в любом случае повторит попытку.

- summarize\_text

```
#!/usr/bin/env python3
import requests
import sys
import json

API_TOKEN = "huggingface"
API_URL = "https://api-inference.huggingface.co/models/facebook/bart-large-cnn" # Модель для резюмирования
headers = {"Authorization": f"Bearer {API_TOKEN}"}
```

# Получение входного текстового файла и имени выходного файла из аргументов командной строки

```
input_file = sys.argv[1]
output_file = sys.argv[2]
```

# Чтение текста из входного файла

```
with open(input_file, 'r') as f:
    text = f.read()
```

# Отправка запроса на API для создания резюме

```
payload = {"inputs": text}
response = requests.post(API_URL, headers=headers, json=payload)
```

# Проверяем, успешно ли выполнен запрос

```
if response.status_code == 200:
    result = response.json()

    # Сохраняем результат в выходной файл
    with open(output_file, 'w') as f:
        for summary in result:
            f.write(summary['summary_text'] + '\n')
else:
    print("Error:", response.text)
    sys.exit(1)
```

Так же используем json и api для отправки серверу запрос, использовал барда для резюмирования. Api снова замазал, ну всё же выкладывать его в открытый гит хаб не хочется.

- save\_to\_pdf

```
#!/usr/bin/env python3

from fpdf import FPDF
import sys

# Получение входного текстового файла и имени выходного PDF-файла из аргументов командной строки
input_file = sys.argv[1]
output_file = sys.argv[2]

# Создание PDF-документа
pdf = FPDF()
pdf.add_page()
pdf.set_font("Arial", size=12)

# Установка отступов
pdf.set_left_margin(20)
pdf.set_right_margin(20)

# Чтение текста из входного файла
with open(input_file, 'r') as f:
    text = f.read()

# Добавление текста в PDF с учетом ширины страницы
# Используем multi_cell для автоматического переноса текста
pdf.multi_cell(0, 10, text)

# Сохранение PDF в выходной файл
pdf.output(output_file)
```

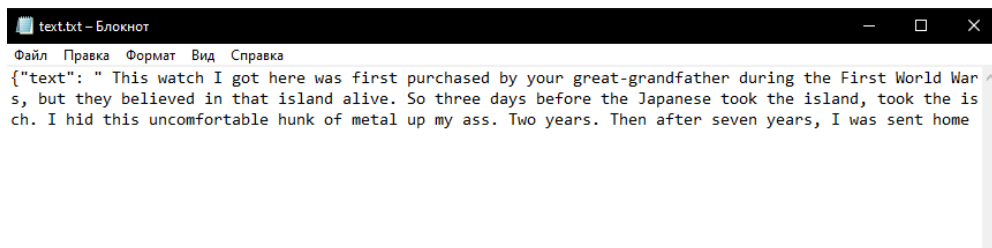
Постарался сделать выходной файл чуть лучше потому, что он выходил одной строкой до этого и упирался куда-то в край листа, решил, что будет лучше использовать `multi_cell` и ограничить лист.

## 7. Тестирование

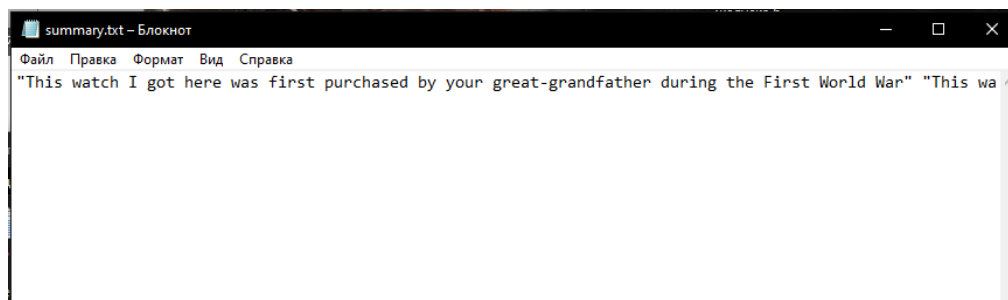
Решил взять легендарный монолог про золотые часы из Криминального чтива.



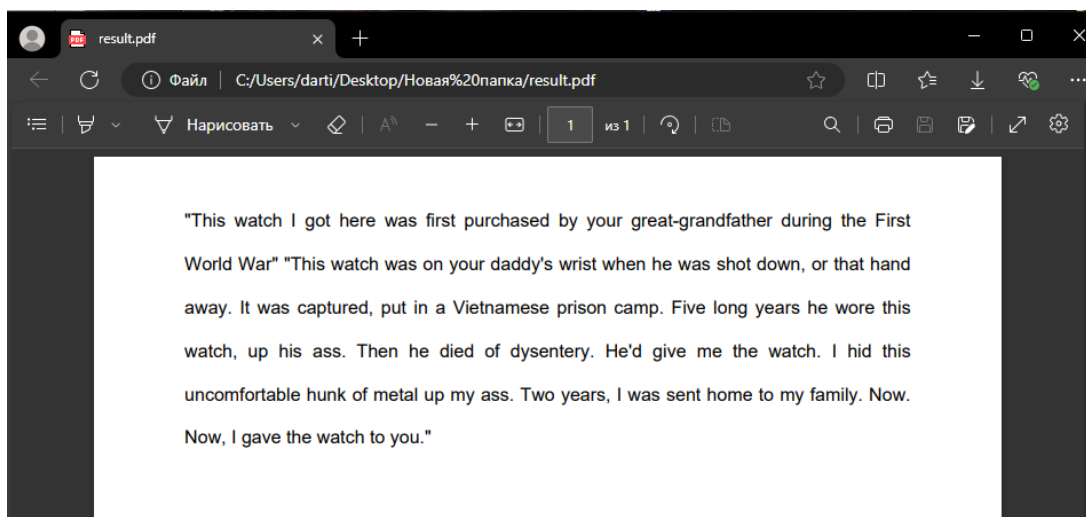
После получил `text.txt`



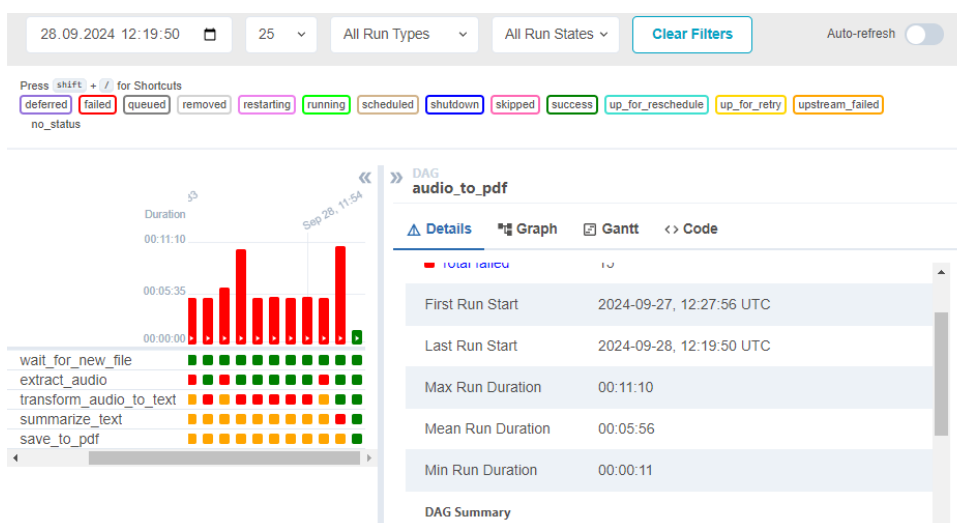
После summary.txt



И наконец финальный результат



Странно, именно так сократил бард, но суть он вроде передал)



Выполненный dag, попыток ушло намеренно, но он заработал.

Docker operator, работал не стабильно, чаще всего после перезапуска контейнера он работал нормально, но не при автоматическом включении airflow.

## Задание 2: Пайплайн для обучения модели.

Я уже создал в прошлом задании контейнер с нужными библиотеками, так что можно использовать его для выполнения задания.

### 1. Разработка DAG

```
dag_lr2_task_2.py > ...
1  from datetime import datetime
2  from airflow import DAG
3  from airflow.providers.docker.operators.docker import DockerOperator
4  from docker.types import Mount # Импортируем Mount
5  from airflow.sensors.filesystem import FileSensor
6
7  default_args = {
8      "owner" : 'darti',
9      'start_date' : datetime(2024, 1, 2),
10     "retries": 1,
11 }
12
13 dag = DAG(
14     'model_learning',
15     default_args = default_args,
16     description='DAG for learn model',
17     schedule_interval=None,
18 )
19
20
21
22 # Задача 1: Загрузка и предобработка данных
23 load_and_preprocess = DockerOperator(
24     task_id='load_and_preprocess_data',
25     image="darti563/my_ml_container",
26     command=["python", "/data/load_and_preprocess_data.py"],
27     mounts=[Mount(source='/data', target='/data', type='bind')],
28     docker_url="tcp://docker-proxy:2375",
29     dag=dag,
30 )
31
32 # Задача 2: Обучение модели
33 load_data_train_model = DockerOperator(
34     task_id = 'load_data_train_model',
35     image = "darti563/my_ml_container",
36     command = ["python", "/data/load_data_train_model.py"],
37     mounts=[Mount(source='/data', target='/data', type='bind')],
38     docker_url="tcp://docker-proxy:2375",
39     dag=dag,
40 )
41
42 load_and_preprocess >> load_data_train_model
```

Dag содержит два task

1. Загрузка и предобработка данных (load\_and\_preprocess\_data): Эта задача запускает контейнер Docker с образом darti563/my\_ml\_container и выполняет скрипт load\_and\_preprocess\_data.py для загрузки и предобработки данных. Данные монтируются в контейнер с локальной директории /data.
2. Обучение модели (load\_data\_train\_model): После завершения первой задачи эта задача запускает другой Docker-контейнер с тем же образом и выполняет скрипт load\_data\_train\_model.py для обучения модели.

## 2. Разработка скрипта для подготовки данных

```
load_and_preprocess_data.py > ...
1  import os
2  import pandas as pd
3  from sklearn.model_selection import train_test_split
4  from sklearn.preprocessing import LabelEncoder
5
6  # Путь к папке с данными
7  data_dir = '/data/lr2'
8  output_dir = '/data/lr2/opr' # Папка для сохранения обработанных данных
9  os.makedirs(output_dir, exist_ok=True) # Создаем папку, если её нет
10
11 def load_and_preprocess_data():
12     # Загружаем все файлы CSV из data_dir и объединяем в один DataFrame
13     data_frames = []
14     for file_name in os.listdir(data_dir):
15         if file_name.endswith('.csv'):
16             file_path = os.path.join(data_dir, file_name)
17             df = pd.read_csv(file_path)
18             data_frames.append(df)
19
20     # Объединяем все данные в один DataFrame
21     full_data = pd.concat(data_frames, ignore_index=True)
22
23     # Кодирование категориальных значений меток (diagnosis)
24     label_encoder = LabelEncoder()
25     full_data['diagnosis'] = label_encoder.fit_transform(full_data['diagnosis'])
26
27     # Разделяем данные на X и y
28     X = full_data.drop(columns=['diagnosis']) # Признаки
29     y = full_data['diagnosis']               # Метки
30
31     # Разделяем данные на обучающую и тестовую выборки
32     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
33
34     # Сохраняем обучающие и тестовые наборы данных
35     X_train.to_csv(os.path.join(output_dir, 'X_train.csv'), index=False)
36     X_test.to_csv(os.path.join(output_dir, 'X_test.csv'), index=False)
37     y_train.to_csv(os.path.join(output_dir, 'y_train.csv'), index=False)
38     y_test.to_csv(os.path.join(output_dir, 'y_test.csv'), index=False)
39
40     print("Предобработанные данные сохранены в папке:", output_dir)
41
42 if __name__ == "__main__":
43     load_and_preprocess_data()
44
```

Данный код выполняет следующие задачи

- Загрузка данных: Все файлы CSV в папке /data/lr2 загружаются и объединяются в один DataFrame.
- Кодирование меток: Категориальные метки в колонке diagnosis кодируются в числовой формат с помощью LabelEncoder.
- Разделение данных: Данные делятся на признаки (X) и метки (y), а затем на обучающую и тестовую выборки с помощью train\_test\_split.
- Сохранение данных: Обучающие и тестовые наборы данных сохраняются как отдельные CSV-файлы в папке /data/lr2/opr.

### 3. Разработка скрипта для создания обучения модели нейронной сети

```
load_data_train_model.py > ...
1 import os
2 import numpy as np
3 import tensorflow as tf
4 from sklearn.model_selection import train_test_split
5 from sklearn.datasets import load_breast_cancer
6 from tensorflow.keras.models import Sequential
7 from tensorflow.keras.layers import Dense
8 from tensorflow.keras.callbacks import CSVLogger
9 import pandas as pd
10
11 # Путь к папке с предварительно обработанными данными
12 data_dir = '/data/lr2/opr'
13
14 # Загрузка данных из файлов
15 X_train = pd.read_csv(os.path.join(data_dir, 'X_train.csv')).values
16 X_test = pd.read_csv(os.path.join(data_dir, 'X_test.csv')).values
17 y_train = pd.read_csv(os.path.join(data_dir, 'y_train.csv')).values.ravel()
18 y_test = pd.read_csv(os.path.join(data_dir, 'y_test.csv')).values.ravel()
19
20 model = Sequential([
21     Dense(30, input_shape=(X_train.shape[1],), activation='relu'), # Входной слой с 30 нейронами
22     Dense(15, activation='relu'), # Скрытый слой с 15 нейронами и функцией активации ReLU
23     Dense(1, activation='sigmoid') # Выходной слой с 1 нейроном и сигмоидальной активацией для бинарной классификации
24 ])
25
26 # Компилируем модель, указывая оптимизатор, функцию потерь и метрики для оценки
27 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
28
29 # Задаем директорию для сохранения логов обучения
30 log_dir = '/data/logs'
31 os.makedirs(log_dir, exist_ok=True)
32
33 # Определяем файл для записи логов обучения и создаем CSV логгер
34 log_file = os.path.join(log_dir, 'training_logs.csv')
35 csv_logger = CSVLogger(log_file, append=True)
36
37 # Тренируем модель на обучающем наборе данных, указывая тестовые данные для проверки
38 # Используем csv_logger для записи метрик в файл на каждой эпохе
39 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=1000, callbacks=[csv_logger])
40
41 # Сохраняем обученную модель в файл в указанной директории
42 model.save(os.path.join(log_dir, 'trained_model.h5'))
43
44 print("Обучение завершено и модель сохранена.")
```

Данный код выполняет следующие задачи:



- 1) Загрузка данных: Чтение предварительно обработанных данных (обучающих и тестовых) из файлов в папке /data/lr2/opr.
- 2) Определение модели: Создание нейронной сети с использованием Keras:
  - Входной слой с 30 нейронами и активацией ReLU.
  - Скрытый слой с 15 нейронами и активацией ReLU.
  - Выходной слой с одним нейроном и сигмоидальной активацией для бинарной классификации.
- 3) Компиляция модели: Настройка модели с оптимизатором adam, функцией потерь binary\_crossentropy и метрикой accuracy.
- 4) Логирование обучения: Создание лог-файла training\_logs.csv в папке /data/logs для записи метрик обучения с помощью CSVLogger.
- 5) Обучение модели: Тренировка модели на 1000 эпох, с использованием тестовых данных для проверки на каждой эпохе. Логи записываются в training\_logs.csv.
- 6) Сохранение модели: После обучения модель сохраняется в файл trained\_model.h5 в папке /data/logs.

## 4. Выполнение dag:

Airflow DAGs Cluster Activity Datasets Security Browse Admin Docs 18:18 UTC AA

06.11.2024 18:10:42 25 All Run Types All Run States Clear Filters Auto-refresh

Press **SHIFT** + **/** for Shortcuts

deferred failed queued removed restarting running scheduled shutdown skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

DAG **model\_learning** Run 2024-11-06, 18:10:41 UTC Task **load\_and\_preprocess\_data**

Clear task Mark state as... Filter Tasks

Details Graph Gantt Code Logs

(by attempts)

1

All Levels All File Sources Wrap Download See More

Duration: 22:04:07:50

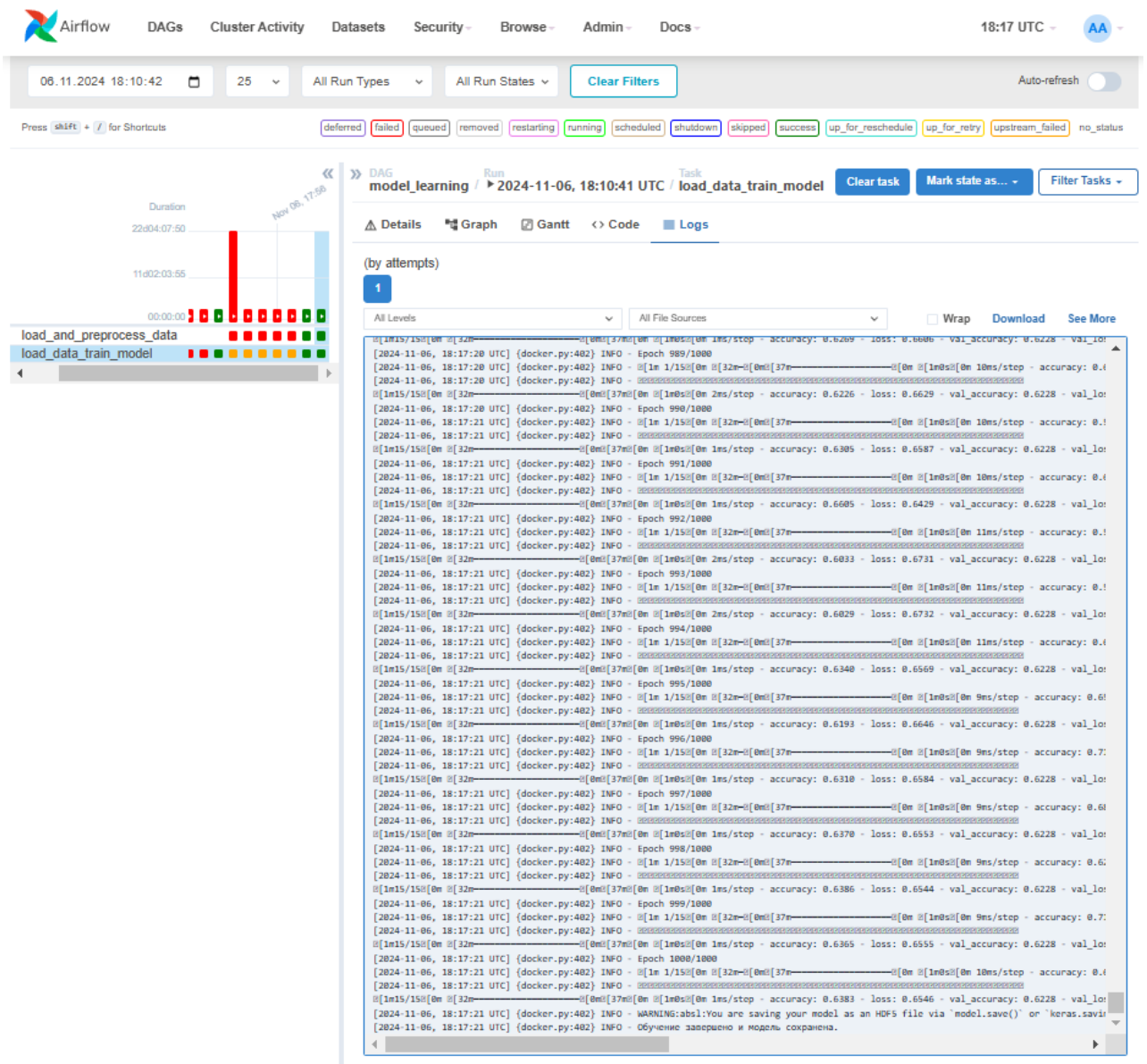
Nov 06, 17:50

00:00:00

load\_and\_preprocess\_data

load\_data\_train\_model

```
[2024-11-06, 18:10:45 UTC] (docker.py:474) INFO - cd4854f88660: Pulling fs layer
[2024-11-06, 18:10:45 UTC] (docker.py:474) INFO - 1a798dfc7d3c: Pulling fs layer
[2024-11-06, 18:10:45 UTC] (docker.py:474) INFO - cb39ec886a1b: Pulling fs layer
[2024-11-06, 18:10:45 UTC] (docker.py:474) INFO - 91bc3f9f8f72: Pulling fs layer
[2024-11-06, 18:10:45 UTC] (docker.py:474) INFO - cb39ec886a1b: Waiting
[2024-11-06, 18:10:45 UTC] (docker.py:474) INFO - 1a798dfc7d3c: Waiting
[2024-11-06, 18:10:45 UTC] (docker.py:474) INFO - 91bc3f9f8f72: Waiting
[2024-11-06, 18:10:45 UTC] (docker.py:474) INFO - cd4854f88660: Waiting
[2024-11-06, 18:10:46 UTC] (docker.py:474) INFO - ddec35dfce011: Downloading
[2024-11-06, 18:10:46 UTC] (docker.py:474) INFO - 8a7a13818b7d: Downloading
[2024-11-06, 18:10:48 UTC] (docker.py:474) INFO - ddec35dfce011: Verifying Checksum
[2024-11-06, 18:10:48 UTC] (docker.py:474) INFO - ddec35dfce011: Download complete
[2024-11-06, 18:10:48 UTC] (docker.py:474) INFO - a2318dc47ec: Downloading
[2024-11-06, 18:10:49 UTC] (docker.py:474) INFO - cd4854f88660: Downloading
[2024-11-06, 18:10:49 UTC] (docker.py:474) INFO - cd4854f88660: Verifying Checksum
[2024-11-06, 18:10:49 UTC] (docker.py:474) INFO - cd4854f88660: Download complete
[2024-11-06, 18:10:50 UTC] (docker.py:474) INFO - 1a798dfc7d3c: Downloading
[2024-11-06, 18:10:50 UTC] (docker.py:474) INFO - 1a798dfc7d3c: Verifying Checksum
[2024-11-06, 18:10:50 UTC] (docker.py:474) INFO - 1a798dfc7d3c: Download complete
[2024-11-06, 18:10:51 UTC] (docker.py:474) INFO - cb39ec886a1b: Downloading
[2024-11-06, 18:10:51 UTC] (docker.py:474) INFO - cb39ec886a1b: Verifying Checksum
[2024-11-06, 18:10:51 UTC] (docker.py:474) INFO - cb39ec886a1b: Download complete
[2024-11-06, 18:10:54 UTC] (docker.py:474) INFO - 91bc3f9f8f72: Downloading
[2024-11-06, 18:10:58 UTC] (docker.py:474) INFO - 8a7a13818b7d: Verifying Checksum
[2024-11-06, 18:10:58 UTC] (docker.py:474) INFO - 8a7a13818b7d: Download complete
[2024-11-06, 18:11:19 UTC] (docker.py:474) INFO - a2318dc47ec: Verifying Checksum
[2024-11-06, 18:11:19 UTC] (docker.py:474) INFO - a2318dc47ec: Download complete
[2024-11-06, 18:11:19 UTC] (docker.py:474) INFO - a2318dc47ec: Extracting
[2024-11-06, 18:11:20 UTC] (docker.py:474) INFO - a2318dc47ec: Pull complete
[2024-11-06, 18:11:20 UTC] (docker.py:474) INFO - ddec35dfce011: Extracting
[2024-11-06, 18:11:20 UTC] (docker.py:474) INFO - ddec35dfce011: Pull complete
[2024-11-06, 18:11:20 UTC] (docker.py:474) INFO - 8a7a13818b7d: Extracting
[2024-11-06, 18:11:20 UTC] (docker.py:474) INFO - 8a7a13818b7d: Pull complete
[2024-11-06, 18:11:20 UTC] (docker.py:474) INFO - cd4854f88660: Extracting
[2024-11-06, 18:11:20 UTC] (docker.py:474) INFO - 1a798dfc7d3c: Extracting
[2024-11-06, 18:11:20 UTC] (docker.py:474) INFO - 1a798dfc7d3c: Pull complete
[2024-11-06, 18:11:20 UTC] (docker.py:474) INFO - cb39ec886a1b: Extracting
[2024-11-06, 18:11:20 UTC] (docker.py:474) INFO - cb39ec886a1b: Pull complete
[2024-11-06, 18:16:23 UTC] (docker.py:474) INFO - 91bc3f9f8f72: Verifying Checksum
[2024-11-06, 18:16:23 UTC] (docker.py:474) INFO - 91bc3f9f8f72: Download complete
[2024-11-06, 18:16:23 UTC] (docker.py:474) INFO - 91bc3f9f8f72: Extracting
[2024-11-06, 18:16:39 UTC] (docker.py:474) INFO - 91bc3f9f8f72: Pull complete
[2024-11-06, 18:16:39 UTC] (docker.py:469) INFO - Digest: sha256:38a84d7eff621177e78832f0f354f359d94fd18ec5af94b49242d892f4e2da5c
[2024-11-06, 18:16:39 UTC] (docker.py:469) INFO - Status: Downloaded newer image for dartis63/my_ml_container:latest
[2024-11-06, 18:16:39 UTC] (docker.py:333) INFO - Starting docker container from image dartis63/my_ml_container
[2024-11-06, 18:16:39 UTC] (docker.py:341) WARNING - Using remote engine or docker-in-docker and mounting temporary volume from host is not
[2024-11-06, 18:16:41 UTC] (docker.py:482) INFO - /pco2o6p607anwz zmwuo coxpanowu n namco: /data/lr2/opr
[2024-11-06, 18:16:42 UTC] (taskinstance.py:1408) INFO - Marking task as SUCCESS. dag_id=model_learning, task_id=load_and_preprocess_data,
[2024-11-06, 18:16:42 UTC] (local_task_job_runner.py:228) INFO - Task exited with return code 0
[2024-11-06, 18:16:42 UTC] (taskinstance.py:2784) INFO - 1 downstream tasks scheduled from follow-on schedule check
```



## 5. Процесс инференса модели

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have ye
WARNING:tensorflow:5 out of the last 9 calls to <function TensorFlowTrai
1/4 ----- 0s 69ms/stepWARNING:tensorflow:6 out of the las
4/4 ----- 0s 16ms/step
Classification Report:
      precision    recall  f1-score   support

   Class 0       0.62       1.00       0.77        71
   Class 1       0.00       0.00       0.00        43

 accuracy         0.62         114
 macro avg       0.31       0.50       0.38       114
 weighted avg    0.39       0.62       0.48       114

Confusion Matrix:
[[71  0]
```

Я решил использовать не написанную мной модель, а модель из sklearn и до

обучить её, так как мной написанная модель, просто выбирала 1 класс и процесса обучения как такового не было(.

Я увеличивал число нейронов, делал dropout и делал балансировку данных, но ничего не сработало, надо было скорее всего брать уже готовую модель, или обучать более чем 1000 эпох.

Возможно стоило взять датасет по проще.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были получены 2 пайплайна:

1. Для работы с llm и серверными ai ботами которые обрабатывают запросы. Использовался hugging face, и в ходе выполнения был получен пайплайн, который обрабатывает входное видео преобразует его в аудио, преобразует аудио в текст, после модель ии сокращает текст, и он сохраняется в pdf.
2. Для предобработки данных и до обучения модели, пайплайн работает, но качество обучения всё же зависит от сложности самой схемы нейронной сети, от данных, которые мы даём ей на обучение и от характеристик самого обучения, возможно я не до обучил модель на 1000 эпох. Но тем не менее модель работает.