Команда: Шмарев М.Ю., Куниевский В.В., Ким К.И., Акимова Д.Д., Рябова Е.Ю.

1. Импортируем все нужные библиотеки для использования нейронной сети.

```
import numpy as np
import pandas as pd
import os
import random
import shutil
import glob
import yaml
import torch
import matplotlib.pyplot as plt
import xml.etree.ElementTree as ET
import cv2

from PIL import Image
from matplotlib import pyplot as plt
from matplotlib import patches
from pathlib import Path
from sklearn.model_selection import train_test_split
```

```
%pip install ultralytics
```

```
[ ] %matplotlib inline
```

для рисования графиков

```
import ultralytics
from ultralytics import YOLO
ultralytics.checks()
```

```
Ultralytics YOLOv8.0.48 🚀 Python-3.8.10 torch-1.13.1+cu116 CPU
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 25.8/107.7 GB disk)
```

Получаем вывод, что работает

2.

```
[ ] !mkdir input
```

```
[ ] directory = "input"
    image_directory = directory + "/images"
    annotation_directory = directory + "/annotations"
    annotations = list(Path(annotation_directory).glob(r'**/*{}'.format('xml')))
```

Пишем пути к файлам, которые требуются для работы в нейронной сети.

3.

```
class_id = {
    "with_mask" : 0,
    "mask_weared_incorrect" : 1,
    "without_mask" : 2
}

data_dict = {
    'filename': [],
    'label': [],
    'class_id': [],
    'width': [],
    'height': [],
    'bboxes': []
}
for annotation_path in annotations:
    tree = ET.parse(annotation_path)
    root = tree.getroot()
    filename = root.find('filename').text
    for obj in root.findall("object"):
        label = obj.find("name").text

        bbox = []
        # bndbox has xmin, ymin, xmax, ymax
        bndbox_tree = obj.find('bndbox')
        bbox.append(int(bndbox_tree.find('xmin').text))
        bbox.append(int(bndbox_tree.find('ymin').text))
        bbox.append(int(bndbox_tree.find('xmax').text))
        bbox.append(int(bndbox_tree.find('ymax').text))
        size = root.find('size')

        data_dict['filename'].append(filename)
        data_dict['width'].append(int(size.find('width').text))
        data_dict['height'].append(int(size.find('height').text))
        data_dict['label'].append(label)
        data_dict['class_id'].append(class_id[label])
        data_dict['bboxes'].append(bbox)

df_data = pd.DataFrame(data_dict)

df_data.head()
```

Прописываем возможные варианты, которые буду получаться и также прописываем вычисления координат для положения масок из их аннотаций.

	filename	label	class_id	width	height	bboxes
0	maksssksksss61.png	with_mask	0	400	273	[3, 151, 51, 234]
1	maksssksksss61.png	without_mask	2	400	273	[162, 9, 178, 27]
2	maksssksksss61.png	with_mask	0	400	273	[117, 7, 141, 26]
3	maksssksksss61.png	with_mask	0	400	273	[11, 1, 33, 12]
4	maksssksksss61.png	with_mask	0	400	273	[93, 37, 114, 59]

Выходные данные

4.

```
train_path = "datasets/train"
valid_path = "datasets/valid"
test_path = "datasets/test"

os.mkdir("datasets")
os.mkdir(train_path)
os.mkdir(valid_path)
os.mkdir(test_path)
```

Создаём папки для тренировки и тестов.

```
train, test = train_test_split(df_data.filename.unique(), test_size=0.2, random_state=23)
train, valid = train_test_split(train, test_size=0.15, random_state=23)

def copy_image_file(image_items, folder_name):
    for image in image_items:
            image_path = image_directory + "/" + image
            new_image_path = os.path.join(folder_name, image)
            shutil.copy(image_path, new_image_path)

def create_label_file(image_items, folder_name):
    for image in image_items:
        fileName = Path(image).stem
        df = df_data[df_data['filename'] == image]
        with open(folder_name + "/" + fileName +'.txt', 'w') as f:
            for i in range(0, len(df)):
                bbox = pascal_voc_to_yolo_bbox(df.iloc[i]['bboxes'], df.iloc[i]['width'], df.iloc[i]['height'])
                bbox_text = " ".join(map(str, bbox))
                txt = str(df.iloc[i]['class_id'])+ " " + bbox_text
                f.write(txt)
                if i != len(df) - 1:
                    f.write("\n")


copy_image_file(train, train_path)
copy_image_file(valid, valid_path)
copy_image_file(test, test_path)

create_label_file(train, train_path)
create_label_file(valid, valid_path)
create_label_file(test, test_path)
```

Перекидываем фотографии в папку “тренировки” и создаём для них описание.

5.

```
classes = list(df_data.label.unique())
class_count = len(classes)
facemask_yaml = f"""
    train: train
    val: valid
    test: test
    nc: {class_count}
    names:
        0 : with_mask
        1 : mask_weared_incorrect
        2 : without_mask
    """

with open('facemask.yaml', 'w') as f:
    f.write(facemask_yaml)

%cat facemask.yaml
```

```
    train: train
    val: valid
    test: test
    nc: 3
    names:
        0 : with_mask
        1 : mask_weared_incorrect
        2 : without_mask
```

Создаем файл yaml, в который записываем данные для нейронной сети.

Посредством данного файла и перенесенных фотографий запускаем обучение на 100 эпох.

```
model = YOLO("yolov8n.pt")
model.train(data="facemask.yaml", epochs=100)
```

```
Validating runs/detect/train2/weights/best.pt...
Ultralytics YOLOv8.0.48 🚀 Python-3.8.10 torch-1.13.1+cu116 CPU
Model summary (fused): 168 layers, 3006233 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:03<00:00,  3.35s/it]
                   all         18         52      0.825      0.572      0.756      0.563
             with_mask         18         43      0.926      0.884      0.941      0.675
 mask_weared_incorrect         18          3          1          0      0.624      0.474
          without_mask         18          6      0.547      0.833      0.704       0.54
Speed: 1.1ms preprocess, 165.0ms inference, 0.0ms loss, 2.4ms postprocess per image
Results saved to runs/detect/train2
```

6.

```
val_label = Image.open("runs/detect/train2/val_batch0_labels.jpg")
val_pred = Image.open("runs/detect/train2/val_batch0_pred.jpg")

plt.figure(figsize=(20,10))
plt.imshow(val_label)
plt.title("Label")
plt.axis(False)
plt.show()

plt.figure(figsize=(20,10))
plt.imshow(val_pred)
plt.title("Prediction")
plt.axis(False)
plt.show()
```

Для проверки работоспособности обученной нейронной сети, показано, как нейронная сеть определяет фотографии, по которым она обучалась.
Label – входные фотографии с аннотациями.
Prediction – результаты работы нейронной сети.

```
predicted_image = Image.open("runs/detect/predict/image0.jpg")
plt.figure(figsize=(10,10))
plt.imshow(predicted_image)
plt.title("Prediction")
plt.axis(False)
plt.show()

predicted_image = Image.open("runs/detect/predict/image1.jpg")
plt.figure(figsize=(10,10))
plt.imshow(predicted_image)
plt.title("Prediction")
plt.axis(False)
plt.show()
```

Также для проверки работоспособности были использованы еще две фотографии. Результат можно увидеть на скриншоте выше.