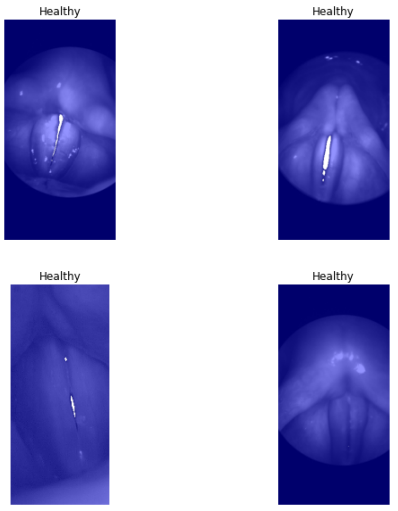
Name:Rutvi shah
Matriculation no.:23159043
Idm:to20raje




```
import os
from PIL import Image
import matplotlib.pyplot as plt

# Function to load an image and its segmentation mask
def load_image_and_mask(image_path, mask_path):
    image = Image.open(image_path)
    mask = Image.open(mask_path)
    return image, mask

# Function to overlay segmentation mask on the image
def overlay_mask(image, mask):
    # Change the RGBA value here (e.g., semi-transparent red overlay)
    overlay = Image.new('RGBA', image.size, (0, 0, 100000000, 108))
    overlay.paste(mask, mask)
    return Image.alpha_composite(image.convert('RGBA'), overlay)

# Path to the folder containing MiniBAGLS dataset
dataset_folder = "C:\\Users\\rutvi\\OneDrive\\Desktop\\Semester 3\\Data science survival skills\\Exercise\\Assignment 3\\Min

# Get a list of image and mask files
image_files = [f for f in os.listdir(dataset_folder) if f.endswith('.png') and not f.endswith('_seg.png')][:4]
mask_files = [f.replace('.png', '_seg.png') for f in image_files]

# Check if there are at least 4 images in the dataset
if len(image_files) < 4:
    print("Not enough images in the dataset.")
    print(f"Found {len(image_files)} image files.")
else:
    # Plotting the first four images with segmentation masks overlaid
    fig, axs = plt.subplots(2, 2, figsize=(10, 10))
```

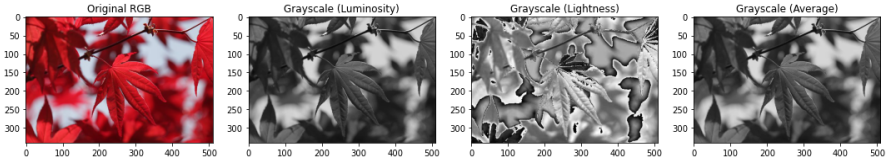
```
for i in range(4):
    image_path = os.path.join(dataset_folder, image_files[i])
    mask_path = os.path.join(dataset_folder, mask_files[i])

    # Extract subject disorder status from the file name
    subject_status = "Healthy" if "healthy" in image_files[i].lower() else "Unhealthy"

    image, mask = load_image_and_mask(image_path, mask_path)
    overlayed_image = overlay_mask(image, mask)

    # Plotting
    row = i // 2
    col = i % 2
    axs[row, col].imshow(overlayed_image)
    axs[row, col].set_title("Healthy")
    axs[row, col].axis('off')

plt.show()
```




```
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np

# Load the RGB image
image_path = "C:\\Users\\rutvi\\OneDrive\\Desktop\\Semester 3\\Data science survival skills\\Exercise\\Assignment 3\\leaves.
rgb_image = Image.open(image_path)

# Convert the RGB image to a NumPy array
rgb_array = np.array(rgb_image)

# Define the luminosity method function
def luminosity_method(rgb):
    return 0.2989 * rgb[:, :, 0] + 0.5870 * rgb[:, :, 1] + 0.1140 * rgb[:, :, 2]

# Define the lightness method function
def lightness_method(rgb):
    return (np.min(rgb, axis=2) + np.max(rgb, axis=2)) / 2

# Define the average method function
def average_method(rgb):
    return np.mean(rgb, axis=2)

# Convert the RGB array to grayscale using different methods
gray_luminosity = luminosity_method(rgb_array)
gray_lightness = lightness_method(rgb_array)
gray_average = average_method(rgb_array)

# Display the original RGB image and the grayscale images side by side
plt.figure(figsize=(15, 4))
```

```
# Original RGB image
plt.subplot(1, 4, 1)
plt.imshow(rgb_array)
plt.title('Original RGB')

# Grayscale using luminosity method
plt.subplot(1, 4, 2)
plt.imshow(gray_luminosity, cmap='gray')
plt.title('Grayscale (Luminosity)')

# Grayscale using lightness method
plt.subplot(1, 4, 3)
plt.imshow(gray_lightness, cmap='gray')
plt.title('Grayscale (Lightness)')

# Grayscale using average method
plt.subplot(1, 4, 4)
plt.imshow(gray_average, cmap='gray')
plt.title('Grayscale (Average)')

# Show the plots
plt.tight_layout()
plt.show()
```

The preferred method for RGB to grayscale conversion depends on the specific use case and the desired perceptual outcome. The Luminosity Method is often preferred because it takes into account the human eye's sensitivity to different color channels, providing a more visually accurate representation of the grayscale image. However, the choice may vary depending on the specific requirements of the application.