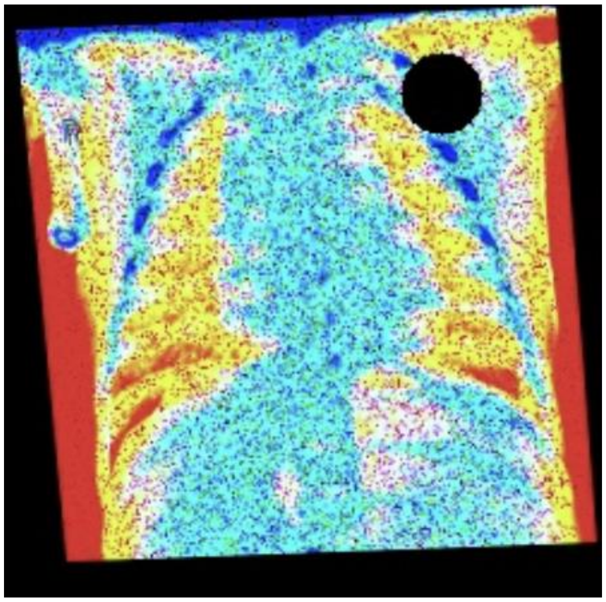
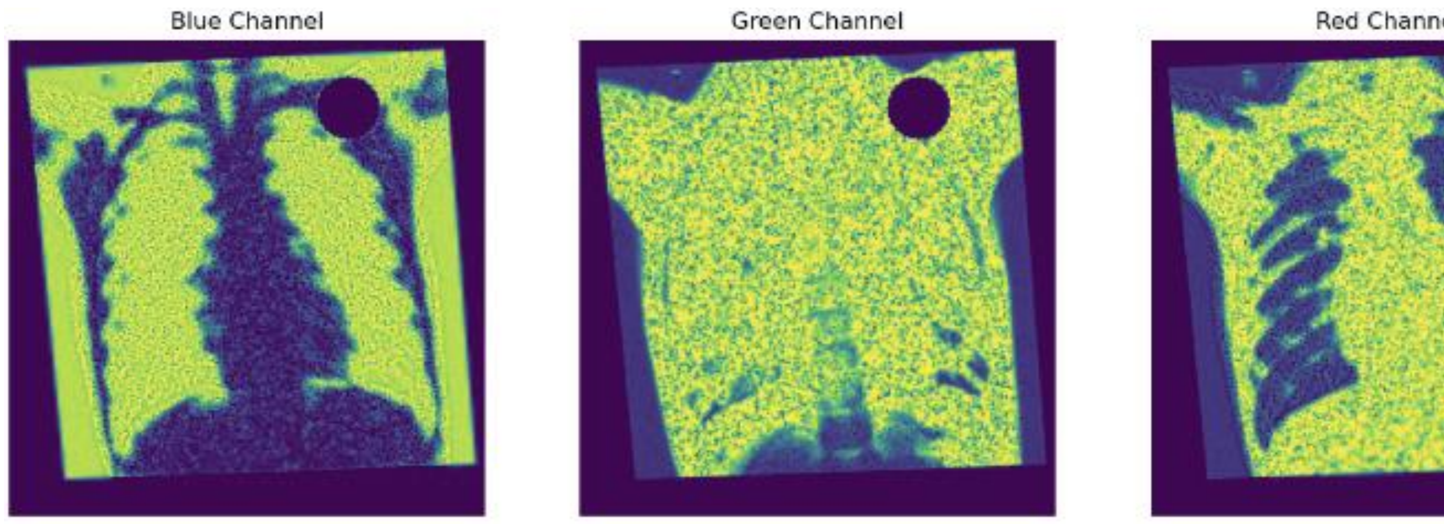
<u>Report on Xray images’ processing</u>

Before applying any processing , we must analyze the images to understand the issues with them and what would be the best way to solve them for better classification of pneumonia.

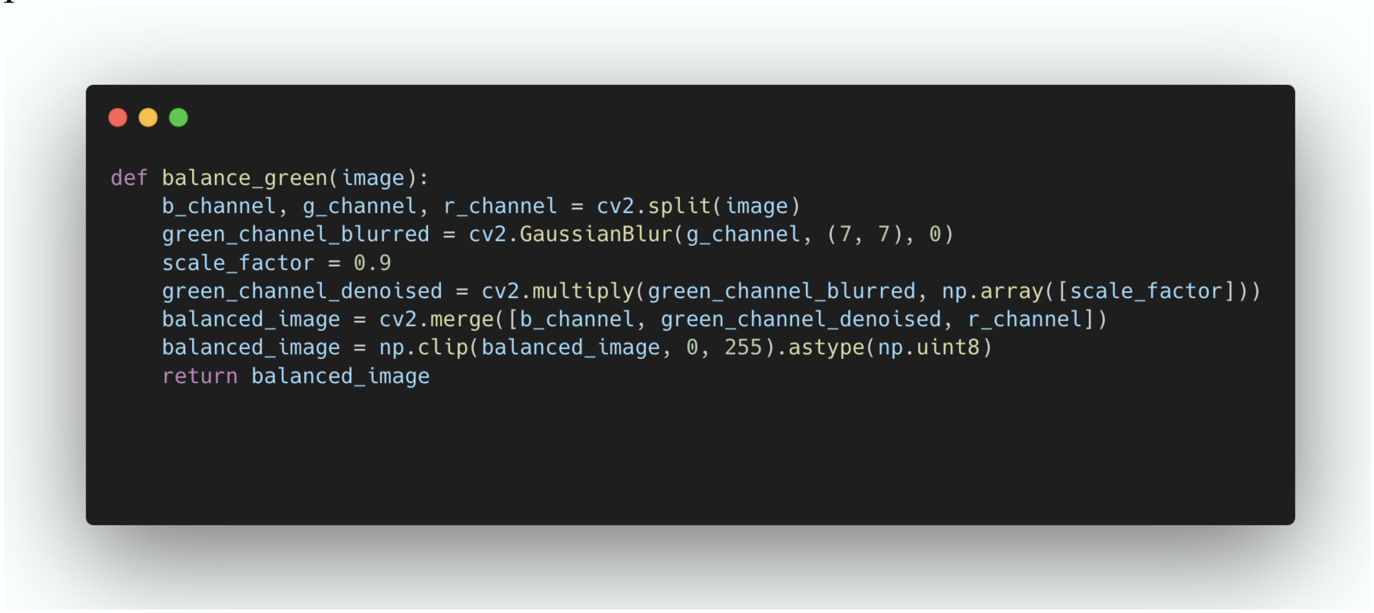


By printing a sample image, we can see the problems in the images , so far it appears to have an unbalancing in the color channels and , there’s alot of noise , the slight rotation to the left and the big black circle .

To understand the unbalance of the colors we show the three-color channels separately :



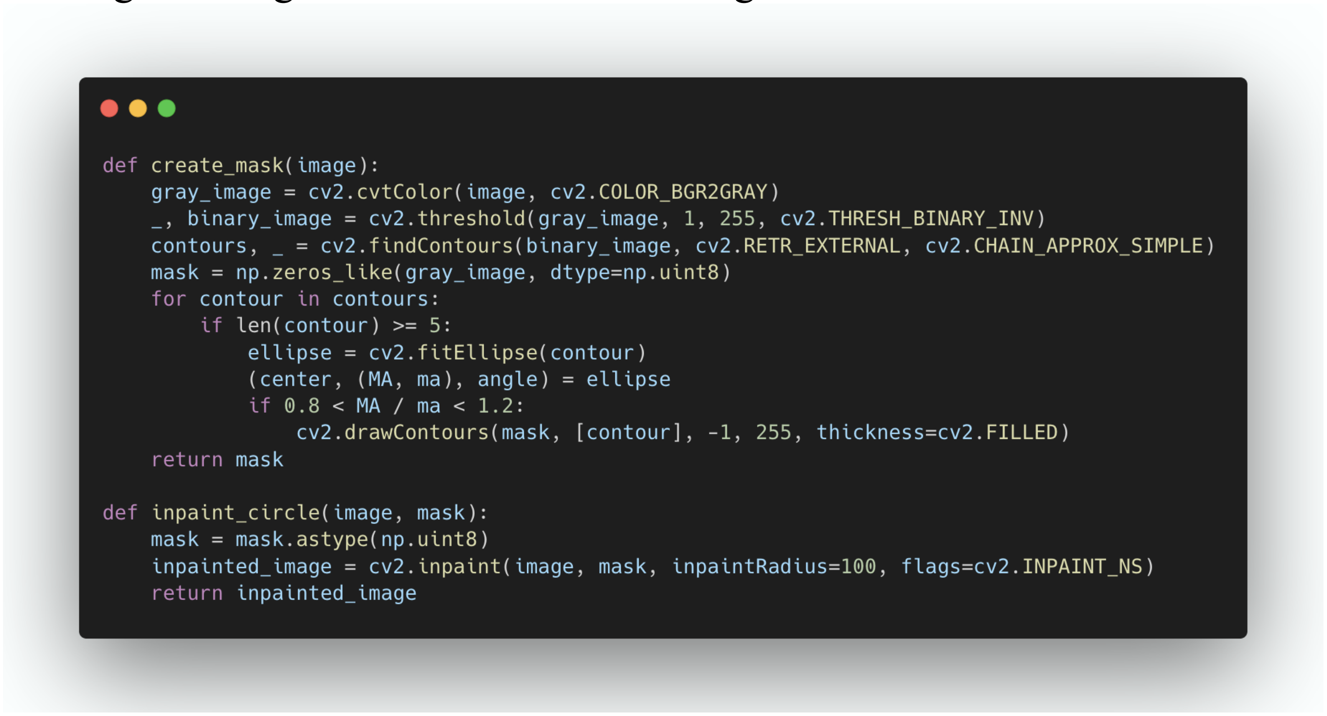
It’s clear that the biggest problem is with the green channel , it has noise the most and very bright compared to the others so we start with this point



```python
def balance_green(image):
    b_channel, g_channel, r_channel = cv2.split(image)
    green_channel_blurred = cv2.GaussianBlur(g_channel, (7, 7), 0)
    scale_factor = 0.9
    green_channel_denoised = cv2.multiply(green_channel_blurred, np.array([scale_factor]))
    balanced_image = cv2.merge([b_channel, green_channel_denoised, r_channel])
    balanced_image = np.clip(balanced_image, 0, 255).astype(np.uint8)
    return balanced_image
```

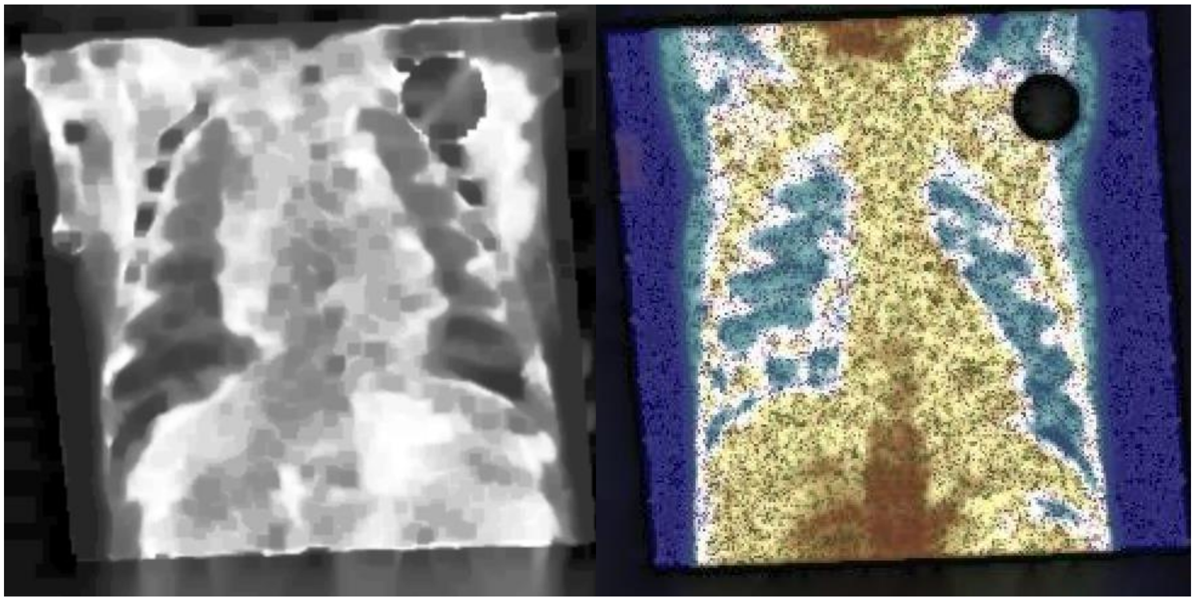
This function isolates the green channel of the image, applies Gaussian blur to reduce noise, and then reduces the intensity by multiplying it by a scale factor (0.9). The blue and red channels are left unchanged. This step is beneficial for noise reduction and slight adjustment of brightness in the green channel, which is be particularly relevant to the observations in the distribution of the colors.

And then for handling the black circle we create two functions , one that is responsible for finding the black dot and the other fills that region in the original image based on the surrounding area.

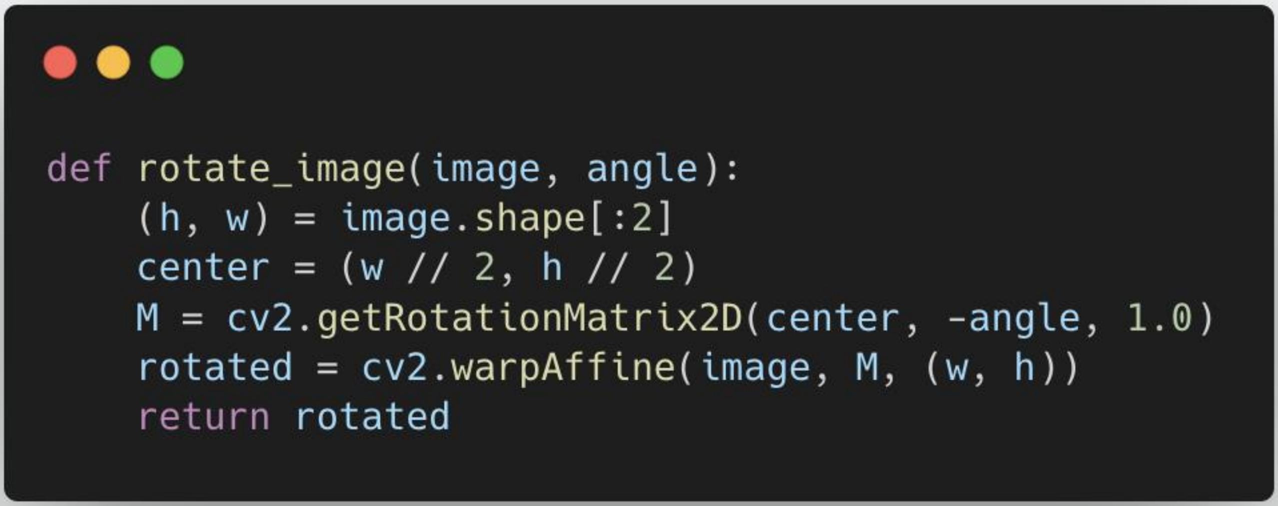


```python
def create_mask(image):
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    _, binary_image = cv2.threshold(gray_image, 1, 255, cv2.THRESH_BINARY_INV)
    contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    mask = np.zeros_like(gray_image, dtype=np.uint8)
    for contour in contours:
        if len(contour) >= 5:
            ellipse = cv2.fitEllipse(contour)
            (center, (MA, ma), angle) = ellipse
            if 0.8 < MA / ma < 1.2:
                cv2.drawContours(mask, [contour], -1, 255, thickness=cv2.FILLED)
    return mask

def inpaint_circle(image, mask):
    mask = mask.astype(np.uint8)
    inpainted_image = cv2.inpaint(image, mask, inpaintRadius=100, flags=cv2.INPAINT_NS)
    return inpainted_image
```

The create_mask(image) function converts the image to grayscale and creates a binary mask that detects and marks bright regions, drawing contours based on detected shapes and focusing on elliptical areas that may be significant in medical images. This mask is later used by the inpaint_circle(image, mask) function to in paint the image, a process that reconstructs lost or deteriorated parts using the cv2.INPAINT_NS method, which is effective for small regions. This inpainting helps clean up the X-ray image by removing or filling in bright areas that could obscure or distort important diagnostic details.


XRAY image after inpainting on grayscale

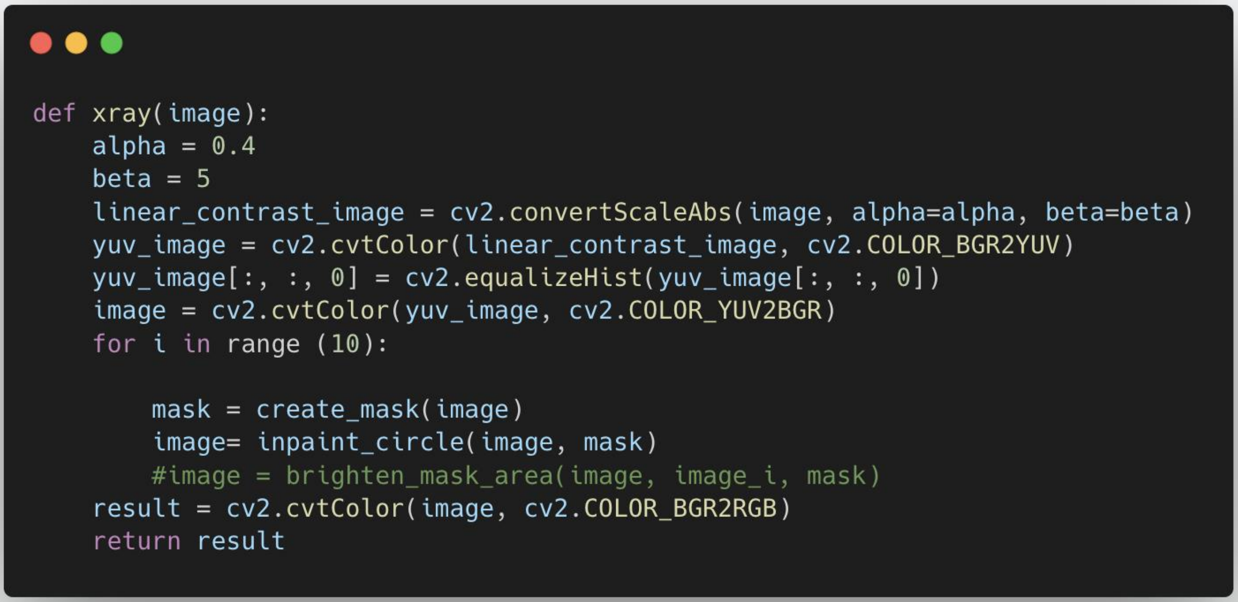
For the slight misalignment we define this function to rotate the image by a specified angle.



```python
def rotate_image(image, angle):
    (h, w) = image.shape[:2]
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, -angle, 1.0)
    rotated = cv2.warpAffine(image, M, (w, h))
    return rotated
```

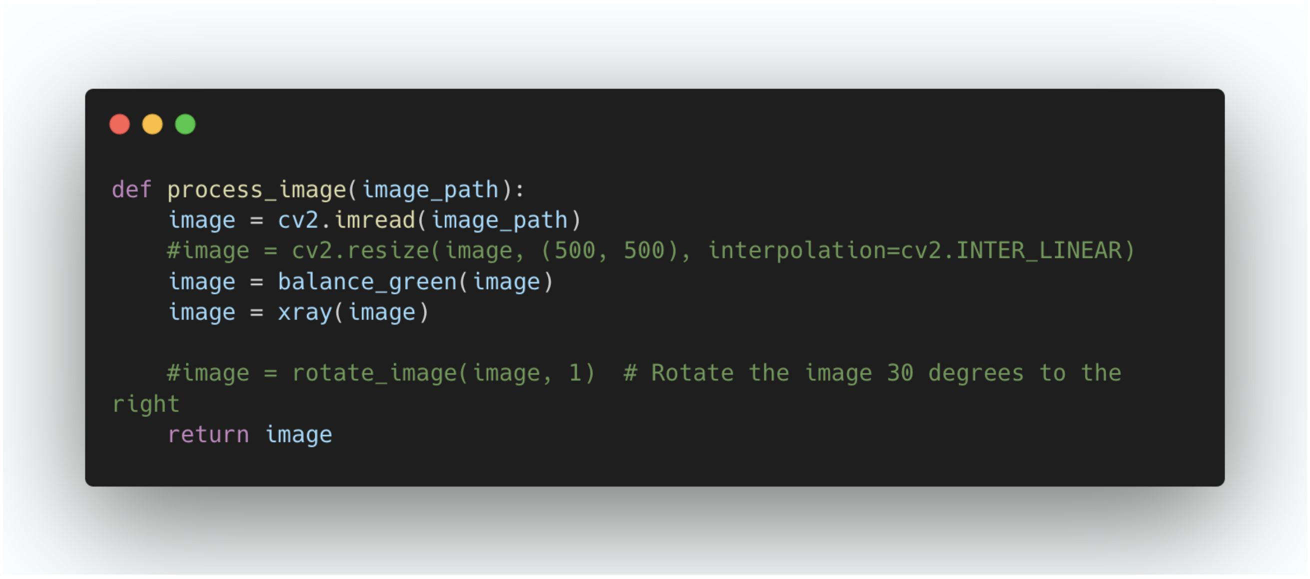
A lot of angle values were tried and the best one was 5 degrees counterclockwise (using -angle) , it helped making the image visually alligned but it didn’t help in the accuracy of the classifier.

To enhance contrast and remove the noise from the image the xray function is used to enhance the images by adjusting contrast, applying histogram equalization, and performing inpainting to improve the visibility of diagnostic features. It begins by enhancing the contrast and brightness using linear scaling and then converts the image to the YUV color space, where histogram equalization is applied to the luminance channel, making details more pronounced. The image is then converted back to the BGR color space, and a mask is created to identify specific regions, such as bright spots that may need correction. Inpainting is applied to these masked areas to fill in or remove artifacts, followed by brightening specific regions to ensure critical areas are clearly visible. Finally, the image is converted to RGB format and returned, resulting in a processed X-ray image with improved clarity and reduced noise, making it more suitable for medical analysis.



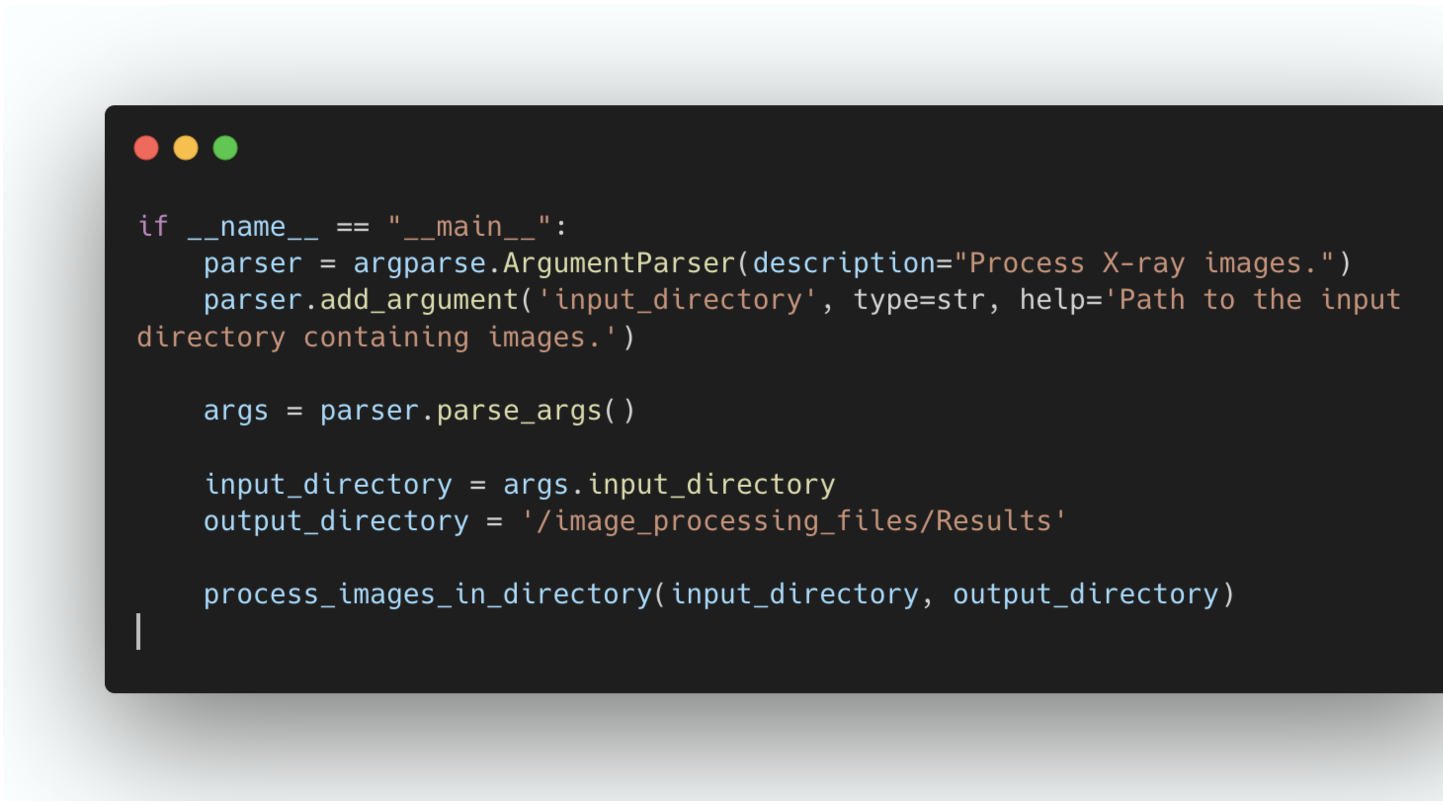
```python
def xray(image):
    alpha = 0.4
    beta = 5
    linear_contrast_image = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
    yuv_image = cv2.cvtColor(linear_contrast_image, cv2.COLOR_BGR2YUV)
    yuv_image[:, :, 0] = cv2.equalizeHist(yuv_image[:, :, 0])
    image = cv2.cvtColor(yuv_image, cv2.COLOR_YUV2BGR)
    for i in range (10):

        mask = create_mask(image)
        image= inpaint_circle(image, mask)
        #image = brighten_mask_area(image, image_i, mask)
    result = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return result
```

Finally we wrap it all in the process image function that reads an image and apply all that was explained before , this helps with the organization of the code and the reuse of all functions to loop and process all images in the directory .



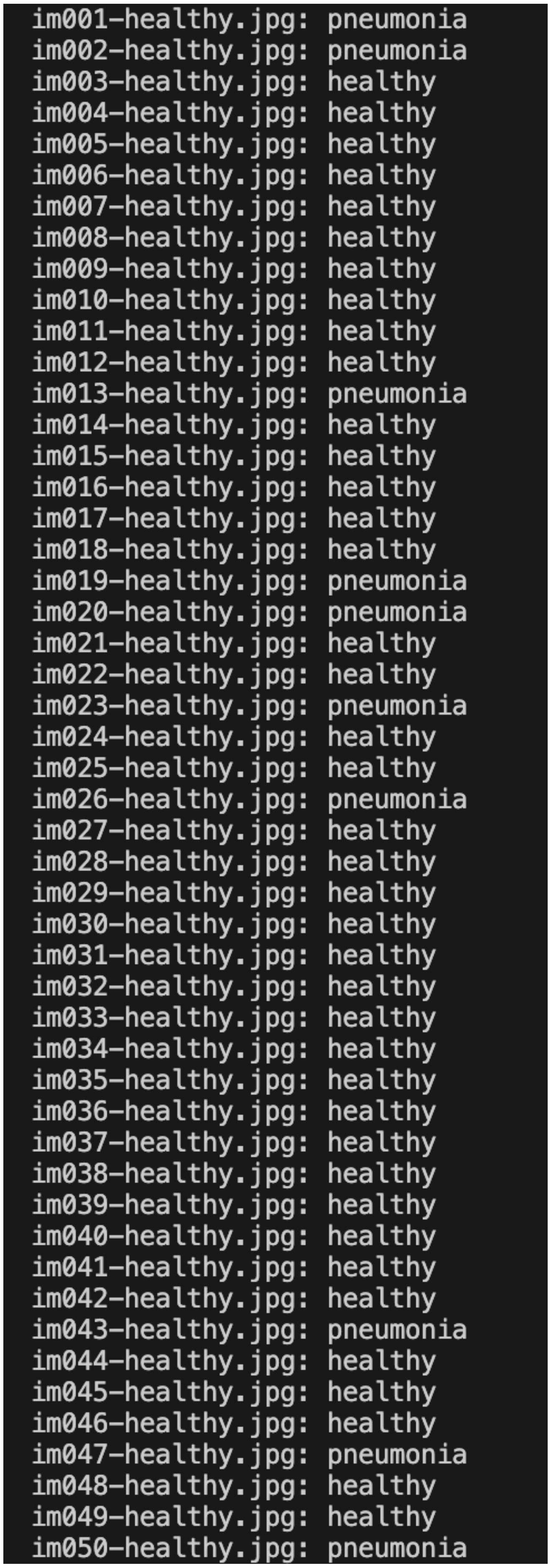
```
def process_image(image_path):
    image = cv2.imread(image_path)
    #image = cv2.resize(image, (500, 500), interpolation=cv2.INTER_LINEAR)
    image = balance_green(image)
    image = xray(image)

    #image = rotate_image(image, 1)  # Rotate the image 30 degrees to the
right
    return image
```

As instructed a main function was implemented to run the code , it takes an input directory and saves the processed images in the output directory named Results

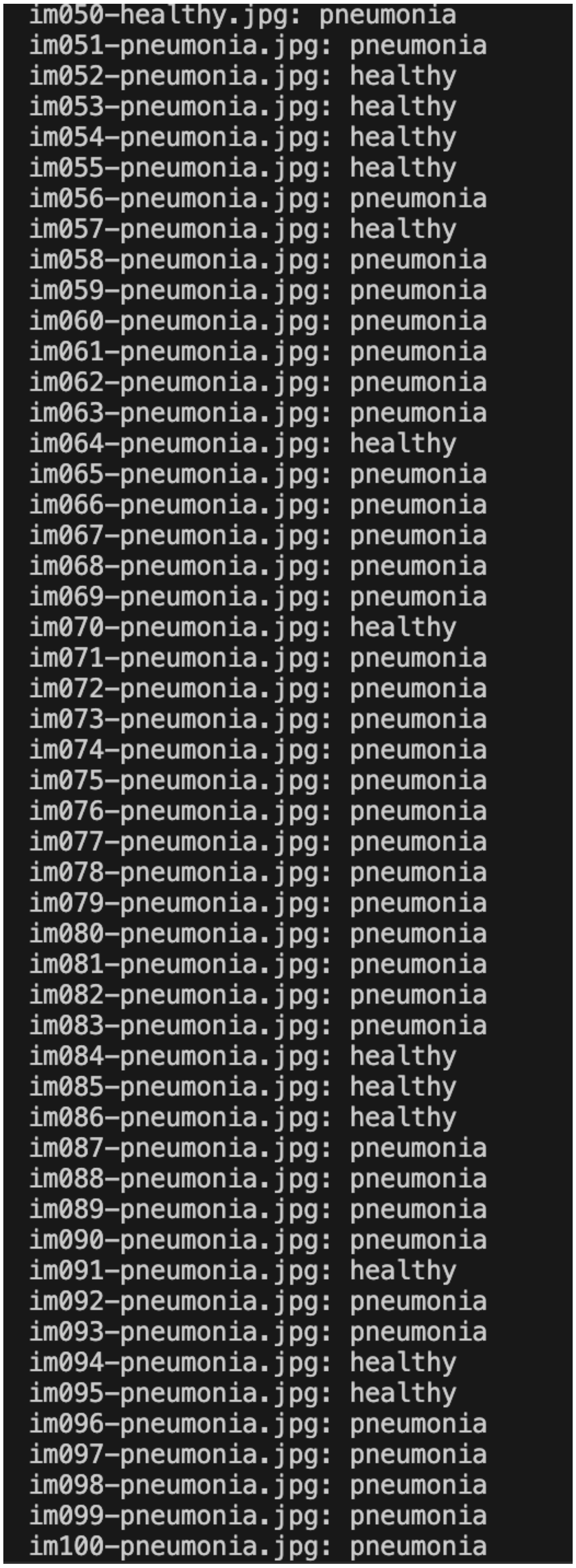


```
if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Process X-ray images.")
    parser.add_argument('input_directory', type=str, help='Path to the input
directory containing images.')

    args = parser.parse_args()

    input_directory = args.input_directory
    output_directory = '/image_processing_files/Results'

    process_images_in_directory(input_directory, output_directory)
```

After running the classify.py that uses the pre_trained model, we got the below results , which are a significant improvement from when the original images were used that gave 55% accuracy.

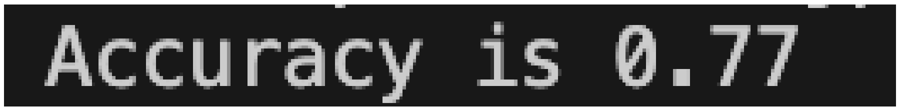


```
im001-healthy.jpg: pneumonia
im002-healthy.jpg: pneumonia
im003-healthy.jpg: healthy
im004-healthy.jpg: healthy
im005-healthy.jpg: healthy
im006-healthy.jpg: healthy
im007-healthy.jpg: healthy
im008-healthy.jpg: healthy
im009-healthy.jpg: healthy
im010-healthy.jpg: healthy
im011-healthy.jpg: healthy
im012-healthy.jpg: healthy
im013-healthy.jpg: pneumonia
im014-healthy.jpg: healthy
im015-healthy.jpg: healthy
im016-healthy.jpg: healthy
im017-healthy.jpg: healthy
im018-healthy.jpg: healthy
im019-healthy.jpg: pneumonia
im020-healthy.jpg: pneumonia
im021-healthy.jpg: healthy
im022-healthy.jpg: healthy
im023-healthy.jpg: pneumonia
im024-healthy.jpg: healthy
im025-healthy.jpg: healthy
im026-healthy.jpg: pneumonia
im027-healthy.jpg: healthy
im028-healthy.jpg: healthy
im029-healthy.jpg: healthy
im030-healthy.jpg: healthy
im031-healthy.jpg: healthy
im032-healthy.jpg: healthy
im033-healthy.jpg: healthy
im034-healthy.jpg: healthy
im035-healthy.jpg: healthy
im036-healthy.jpg: healthy
im037-healthy.jpg: healthy
im038-healthy.jpg: healthy
im039-healthy.jpg: healthy
im040-healthy.jpg: healthy
im041-healthy.jpg: healthy
im042-healthy.jpg: healthy
im043-healthy.jpg: pneumonia
im044-healthy.jpg: healthy
im045-healthy.jpg: healthy
im046-healthy.jpg: healthy
im047-healthy.jpg: pneumonia
im048-healthy.jpg: healthy
im049-healthy.jpg: healthy
im050-healthy.jpg: pneumonia
```



```
im050-healthy.jpg: pneumonia
im051-pneumonia.jpg: pneumonia
im052-pneumonia.jpg: healthy
im053-pneumonia.jpg: healthy
im054-pneumonia.jpg: healthy
im055-pneumonia.jpg: healthy
im056-pneumonia.jpg: pneumonia
im057-pneumonia.jpg: healthy
im058-pneumonia.jpg: pneumonia
im059-pneumonia.jpg: pneumonia
im060-pneumonia.jpg: pneumonia
im061-pneumonia.jpg: pneumonia
im062-pneumonia.jpg: pneumonia
im063-pneumonia.jpg: pneumonia
im064-pneumonia.jpg: healthy
im065-pneumonia.jpg: pneumonia
im066-pneumonia.jpg: pneumonia
im067-pneumonia.jpg: pneumonia
im068-pneumonia.jpg: pneumonia
im069-pneumonia.jpg: pneumonia
im070-pneumonia.jpg: healthy
im071-pneumonia.jpg: pneumonia
im072-pneumonia.jpg: pneumonia
im073-pneumonia.jpg: pneumonia
im074-pneumonia.jpg: pneumonia
im075-pneumonia.jpg: pneumonia
im076-pneumonia.jpg: pneumonia
im077-pneumonia.jpg: pneumonia
im078-pneumonia.jpg: pneumonia
im079-pneumonia.jpg: pneumonia
im080-pneumonia.jpg: pneumonia
im081-pneumonia.jpg: pneumonia
im082-pneumonia.jpg: pneumonia
im083-pneumonia.jpg: pneumonia
im084-pneumonia.jpg: healthy
im085-pneumonia.jpg: healthy
im086-pneumonia.jpg: healthy
im087-pneumonia.jpg: pneumonia
im088-pneumonia.jpg: pneumonia
im089-pneumonia.jpg: pneumonia
im090-pneumonia.jpg: pneumonia
im091-pneumonia.jpg: healthy
im092-pneumonia.jpg: pneumonia
im093-pneumonia.jpg: pneumonia
im094-pneumonia.jpg: healthy
im095-pneumonia.jpg: healthy
im096-pneumonia.jpg: pneumonia
im097-pneumonia.jpg: pneumonia
im098-pneumonia.jpg: pneumonia
im099-pneumonia.jpg: pneumonia
im100-pneumonia.jpg: pneumonia
```

After the image processing we reach 77% accuracy



```
Accuracy is 0.77
```