Experiment:1.1

Aim:

Write a program to implement various feature extraction techniques for image classification.

Software Required:

Any IDE (Jupyter Notebook, Pycharm, Google Colab).

Description:

There are several feature extraction techniques commonly used in image classification tasks. These techniques aim to capture relevant information from images and transform them into meaningful representations that can be used by machine learning algorithms for classification. Some popular feature extraction techniques are:

- **Scale-Invariant Feature Transform (SIFT):** SIFT is a widely used technique that identifies key points and extracts local invariant descriptors from images. It is robust to changes in scale, rotation, and illumination.
- **Speeded-Up Robust Features (SURF):** SURF is another technique that detects and describes local features in images. It is similar to SIFT but computationally more efficient, making it suitable for real-time applications.
- **Histogram of Oriented Gradients (HOG):** HOG computes the distribution of gradient orientations in an image. It captures the shape and edge information and has been particularly successful in object detection and pedestrian recognition tasks.
- **Convolutional Neural Networks (CNN):** CNNs are a type of deep learning model that automatically learn hierarchical features from images. They consist of multiple convolutional layers that extract low-level to high-level features. CNNs have revolutionized image classification and achieved state-of-the-art performance in varioustasks.
- **Color Histograms:** Color histograms capture the distribution of colors in an image. They represent the color content of images by quantizing pixel colors into bins and counting their occurrences. Color histograms are simple yet effective features for certaintypes of image classification problems.
- **Local Binary Patterns (LBP):** LBP encodes the texture information by comparing each pixel's intensity value with its neighboring pixels. It is commonly used in textureanalysis tasks and has shown good performance in various image classification applications.
- **Gabor Filters:** Gabor filters are a set of linear filters that capture localized frequency and

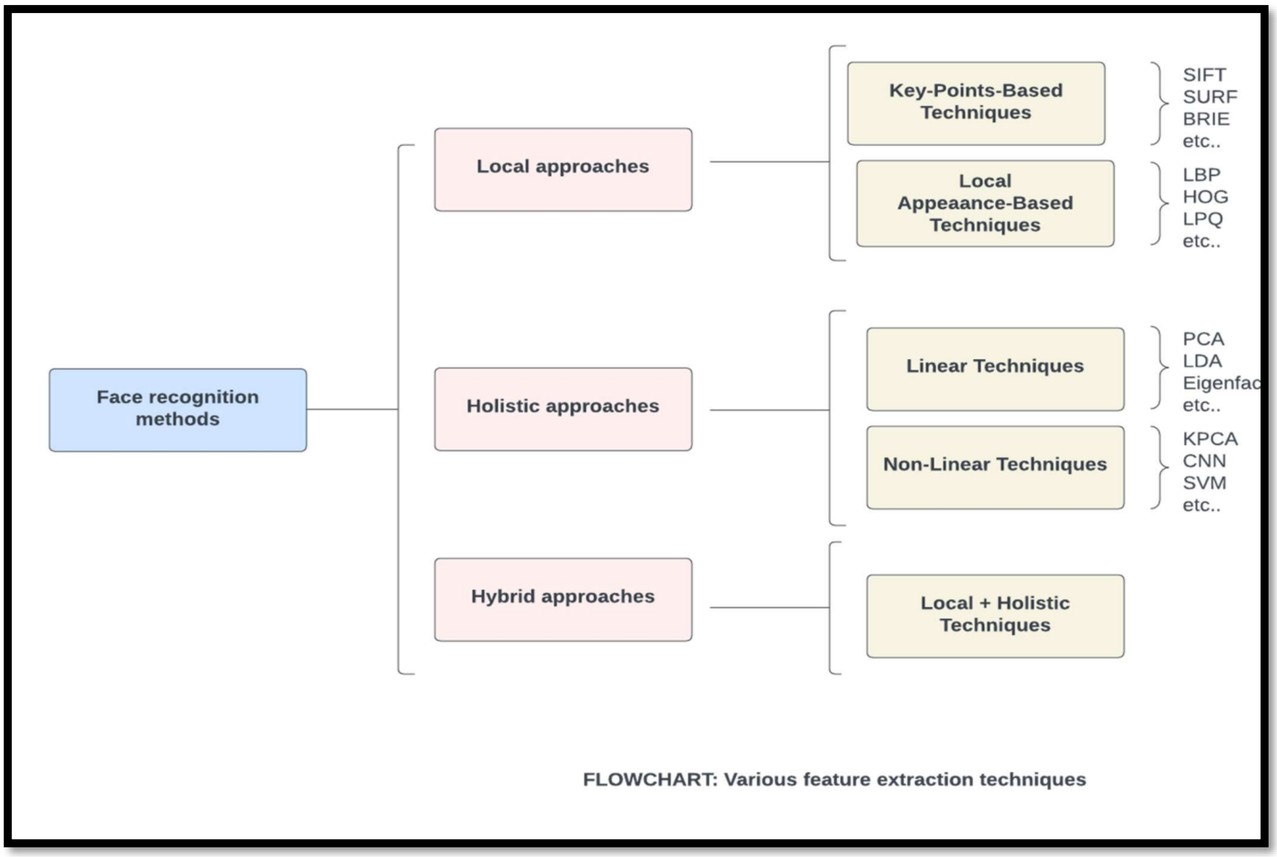
orientation information in images. They are commonly used for texture analysis andhave been successfully applied in face recognition and fingerprint recognition tasks.

- **Deep Convolutional Features:** Instead of using pre-defined feature extraction techniques, it is also common to use pre-trained CNN models (e.g., VGG, ResNet, Inception) and extract features from intermediate layers. These deep convolutional features retain more high-level semantics and have been shown to generalize well acrossdifferent image classification tasks.

The experiment involves implementing different feature extraction techniques for image classification using Python and relevant libraries. The steps involved in this experiment are as follows:

1. Import necessary libraries (e.g., OpenCV, scikit-image).
2. Load the dataset of labeled images for training and testing.
3. Preprocess the images by resizing, normalizing, or applying any necessary transformations.
4. Extract features from the images using various techniques such as:
   - Histogram of Oriented Gradients (HOG)
   - Scale-Invariant Feature Transform (SIFT)
   - Speeded-Up Robust Features (SURF)
   - Local Binary Patterns (LBP)
   - Convolutional Neural Networks (CNN), etc.
5. Split the dataset into training and testing sets.
6. Train a classifier (e.g., Support Vector Machine, Random Forest, etc.) using theextracted features and the corresponding labels.
7. Evaluate the performance of the classifier on the testing set by calculating metrics like accuracy, precision, recall, and F1-score.
8. Compare the performance of different feature extraction techniques by analyzing the evaluation results.
9. Repeat steps 4-8 for different combinations of feature extraction techniques andclassifiers to explore the impact on classification performance.
10. Document the observations and conclusions drawn from the experiment.

Pseudo code/Algorithms/Flowchart/Steps:



Implementation:

```
from google.colab import drive
drive.mount('/content/gdrive')


"""**Import necessary libraries**"""
import cv2 as cv
import skimage.io as io
import matplotlib.pyplot as plt
import skimage.transform as transform
import skimage.segmentation as segmentation
```

```
"""**Load the image**"""
image = io.imread('/content/gdrive/MyDrive/Data/images/Vivek.png')

"""**Display the original Dimension & images**"""
print('Original Dimensions : ', image.shape)
io.imshow(image)

"""**Using Transform the Image is resized**"""
resized_image = transform.resize(image, (200, 200))
print('Resized Dimensions : ', resized_image.shape)
io.imshow(resized_image)

"""**Show Segmented Part of Image**"""
segmented_image = segmentation.slic(image, n_segments=100)
io.imshow(segmented_image)

"""**Assign Plotting Style**"""
plt.style.use('seaborn')

"""**Convert The Images in Different Category**"""
loaded_image = cv.cvtColor(image,cv.COLOR_BGR2RGB)
gray_image = cv.cvtColor(loaded_image,cv.COLOR_BGR2GRAY)
edged_image = cv.Canny(gray_image, threshold1=30, threshold2=100)
```

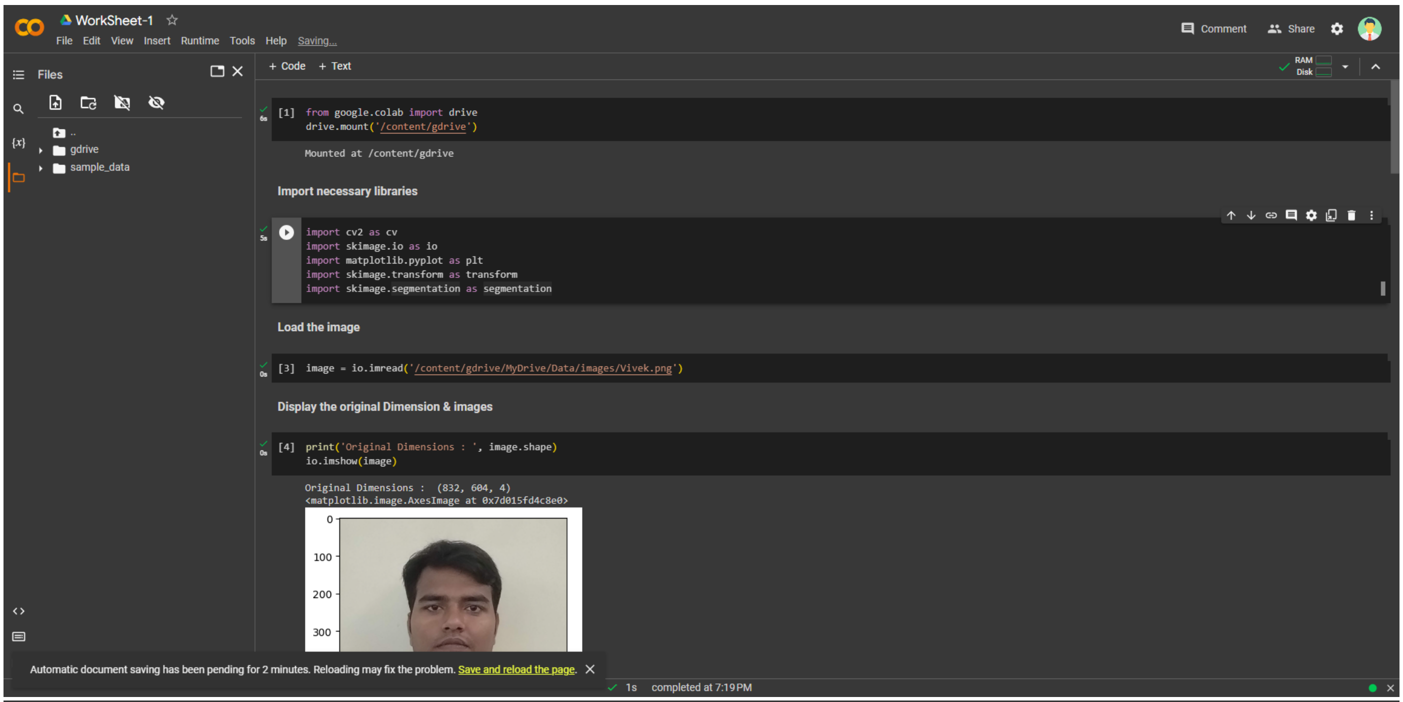
```
"""**Plot the RGB Collered Image **"""
plt.title("RGB Collered Image")
plt.axis("off")
plt.imshow(loaded_image)

"""**Plot the GrayScale Image **"""
plt.title("GrayScale Image")
plt.axis("off")
plt.imshow(gray_image,cmap="gray")

"""**Plot the Canny Edge Detected Image **"""
plt.title("Canny Edge Detected Image")
plt.axis("off")
plt.imshow(edged_image)
```

CourseName: Computer Vision Lab **Course Code:** CSP-422

Output:



```
[1] from google.colab import drive
    drive.mount('/content/gdrive')

    Mounted at /content/gdrive
```

Import necessary libraries

```
import cv2 as cv
import skimage.io as io
import matplotlib.pyplot as plt
import skimage.transform as transform
import skimage.segmentation as segmentation
```

Load the image

```
[3] image = io.imread('/content/gdrive/MyDrive/Data/images/Vivek.png')
```

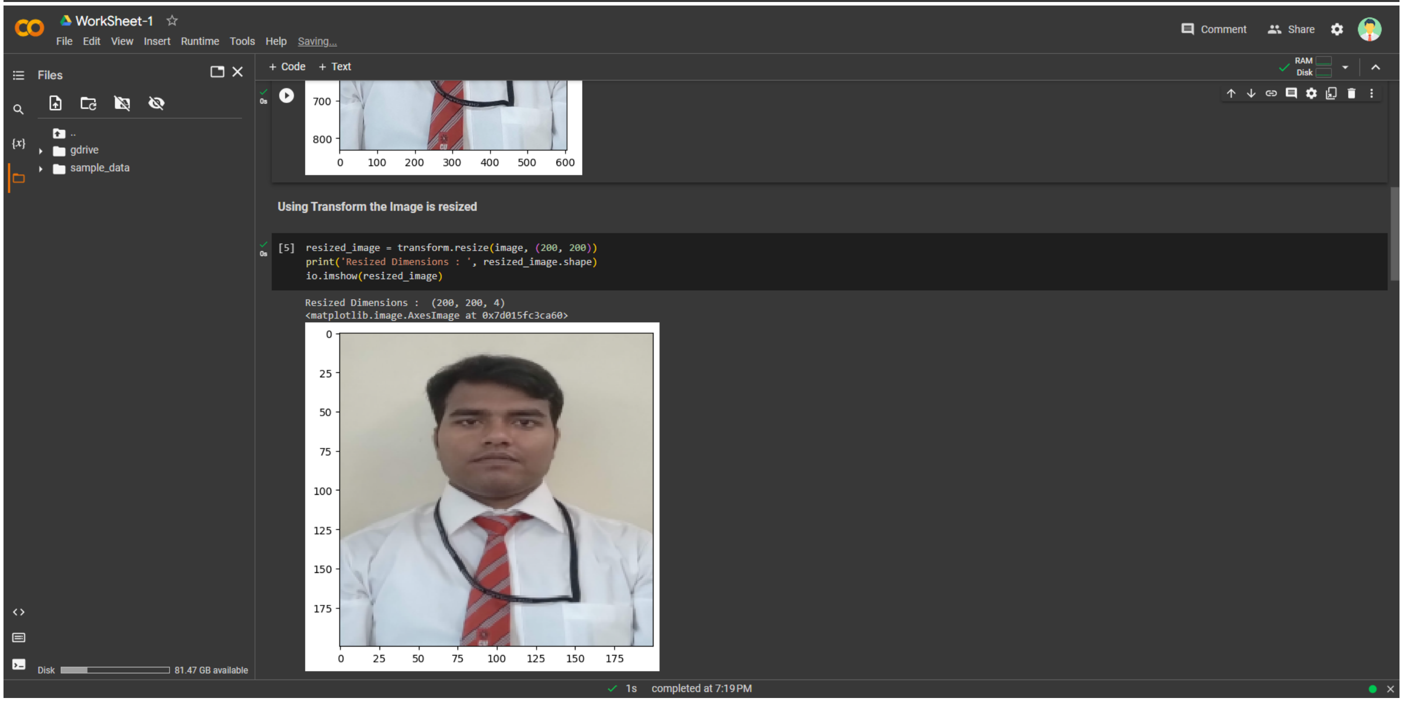
Display the original Dimension & images

```
[4] print('Original Dimensions : ', image.shape)
    io.imshow(image)

    Original Dimensions :  (832, 604, 4)
    <matplotlib.image.AxesImage at 0x7d015fd4c8e0>
```

Automatic document saving has been pending for 2 minutes. Reloading may fix the problem. Save and reload the page.

✓ 1s completed at 7:19 PM



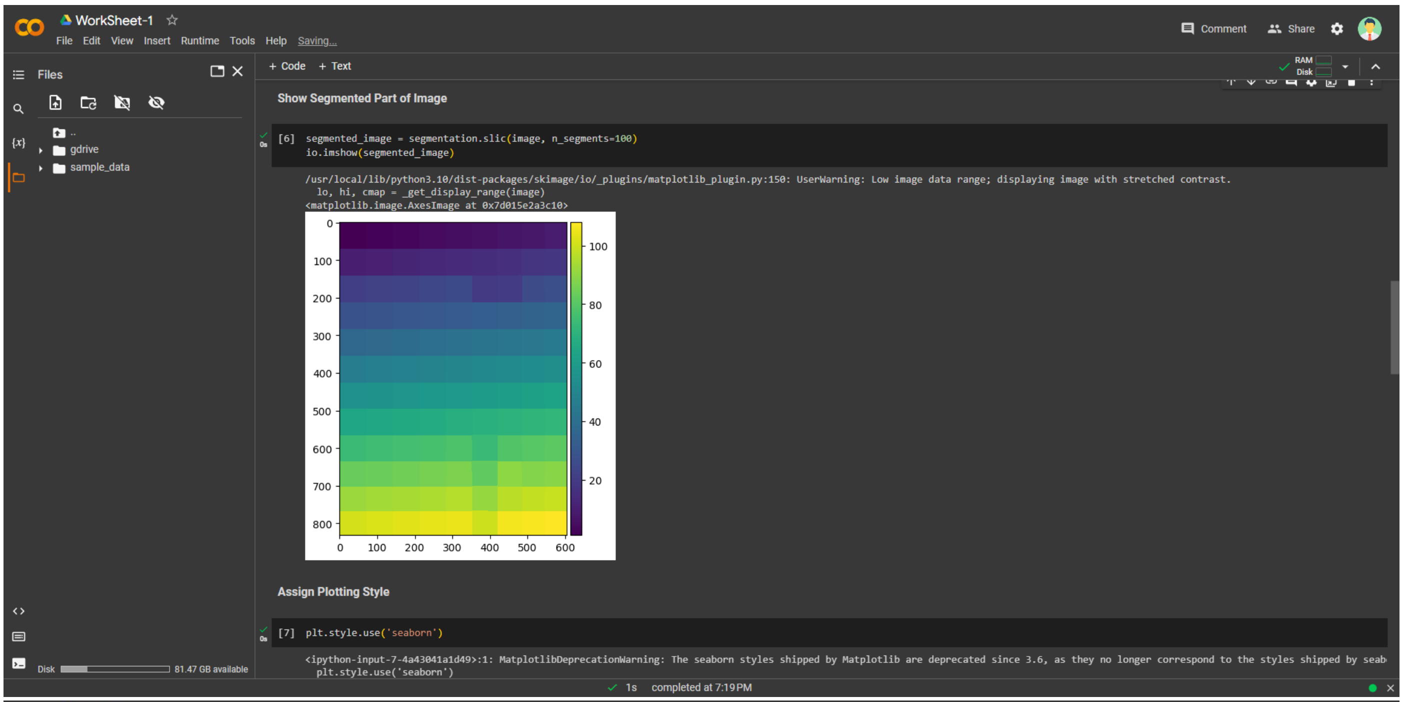
Using Transform the Image is resized

```
[5] resized_image = transform.resize(image, (200, 200))
    print('Resized Dimensions : ', resized_image.shape)
    io.imshow(resized_image)

    Resized Dimensions :  (200, 200, 4)
    <matplotlib.image.AxesImage at 0x7d015fc3ca60>
```

✓ 1s completed at 7:19 PM

CourseName: Computer Vision Lab **Course Code:** CSP-422



Show Segmented Part of Image

```
[6] segmented_image = segmentation.slic(image, n_segments=100)
    io.imshow(segmented_image)
```

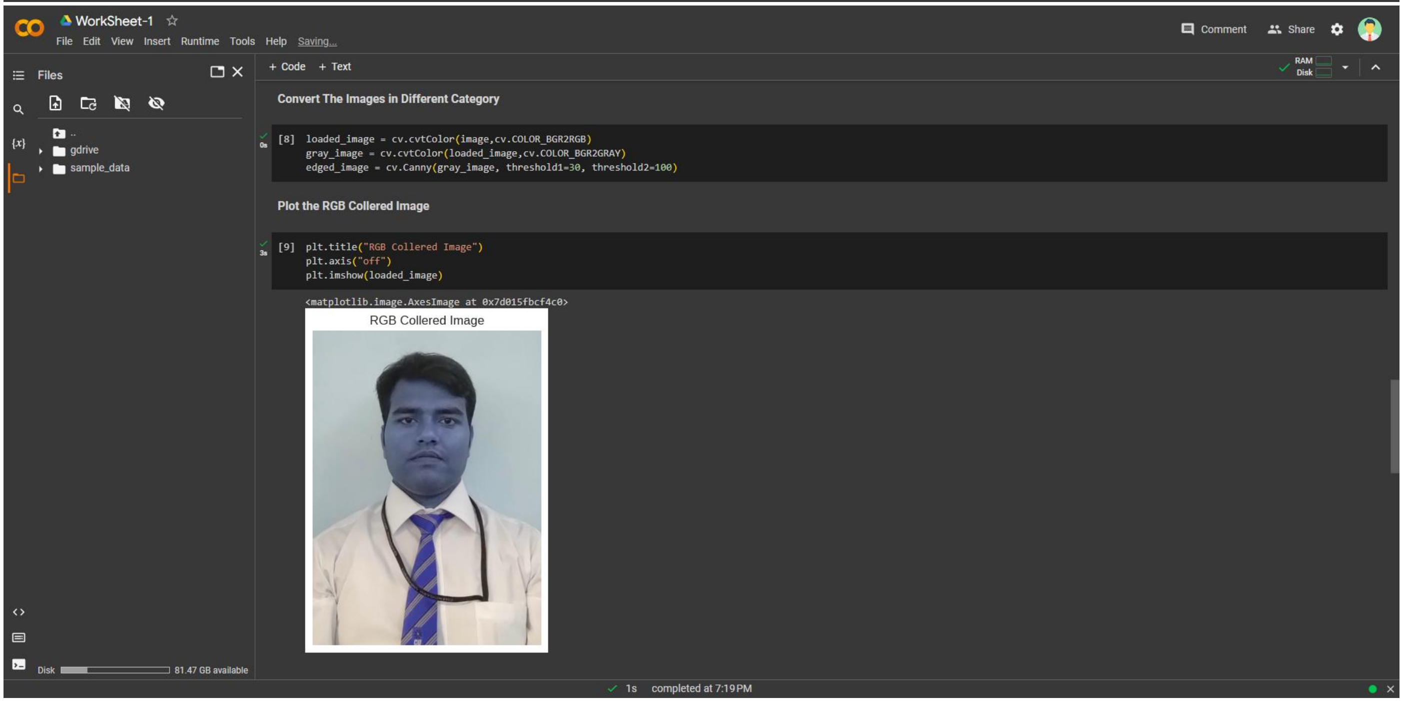
```
/usr/local/lib/python3.10/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
<matplotlib.image.AxesImage at 0x7d015e2a3c10>
```

Assign Plotting Style

```
[7] plt.style.use('seaborn')
```

```
<ipython-input-7-4a43041a1d49>:1: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seab
  plt.style.use('seaborn')
```

✓ 1s completed at 7:19 PM


WorkSheet-1
File Edit View Insert Runtime Tools Help Saving...
Comment Share
Files
gdrive
sample_data
+ Code + Text
Disk 81.47 GB available

Convert The Images in Different Category

```
[8] loaded_image = cv.cvtColor(image,cv.COLOR_BGR2RGB)
    gray_image = cv.cvtColor(loaded_image,cv.COLOR_BGR2GRAY)
    edged_image = cv.Canny(gray_image, threshold1=30, threshold2=100)
```

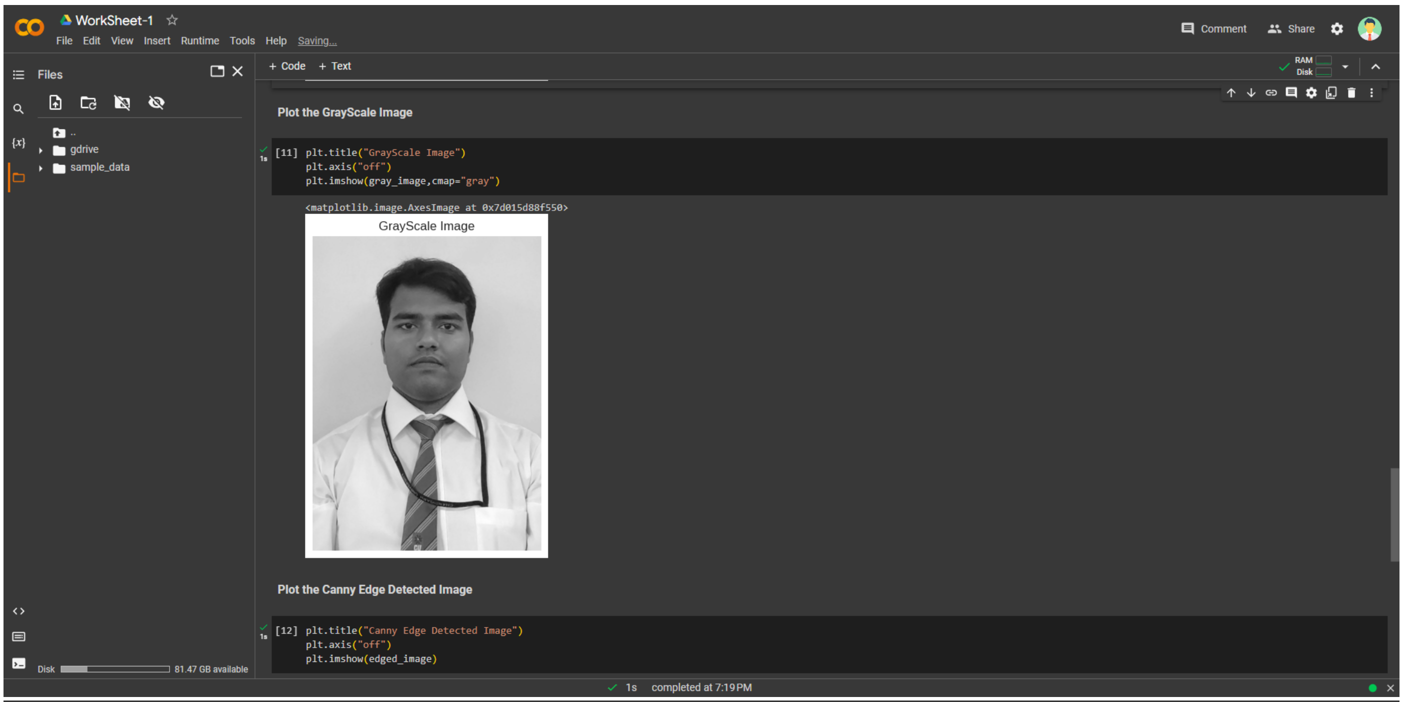
Plot the RGB Collered Image

```
[9] plt.title("RGB Collered Image")
    plt.axis("off")
    plt.imshow(loaded_image)
```

```
<matplotlib.image.AxesImage at 0x7d015fbcf4c0>
```

RGB Collered Image
✓ 1s completed at 7:19 PM

CourseName: Computer Vision Lab **Course Code:** CSP-422



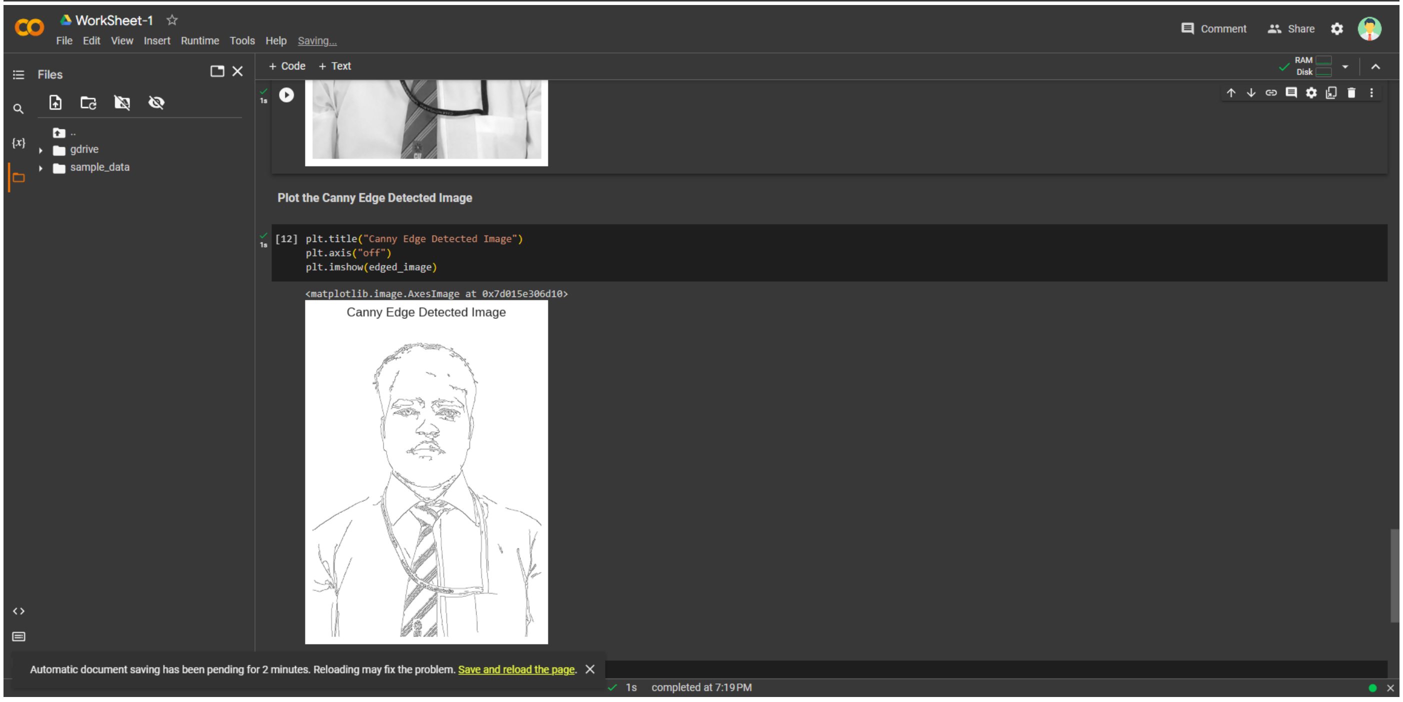
Plot the GrayScale Image

```
[11] plt.title("GrayScale Image")
     plt.axis("off")
     plt.imshow(gray_image,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7d015d88f550>
```

Plot the Canny Edge Detected Image

```
[12] plt.title("Canny Edge Detected Image")
     plt.axis("off")
     plt.imshow(edged_image)
```



Plot the Canny Edge Detected Image

```
[12] plt.title("Canny Edge Detected Image")
     plt.axis("off")
     plt.imshow(edged_image)
```

```
<matplotlib.image.AxesImage at 0x7d015e306d10>
```

Automatic document saving has been pending for 2 minutes. Reloading may fix the problem. Save and reload the page.

WorkSheet-1
File Edit View Insert Runtime Tools Help Saving...
Files
gdrive
sample_data
Canny Edge Detected Image
1s completed at 7:19 PM