**PROJECT Submitted**

**For ARTIFICIAL INTELLIGENCE (UCS414)**

**By**

| | |
|---|---|
| **Jaskiran Kaur Walia** | **102283023** |
| **Varshitha Pentu** | **102103107** |
| **Lavisha Lakhmani** | **102103109** |
| **Shaina** | **102103118** |

**Submitted to**

**Dr. Navpreet Kaur**

# Name of the project :Image processing and filtering

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, (A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB**

## INTRODUCTION

- AI image filtering is a technology that uses artificial intelligence (AI) algorithms to automatically filter or analyze photographs. The idea is to train an artificial intelligence system to analyse and edit photos based on predefined criteria such as image quality, image content, and image features.

- AI image filtering techniques include picture categorization, object identification, image segmentation, and image enhancement. Image classification entails classifying photographs into distinct groups, such as identifying objects, animals, or landscapes.

## LITERATURE REVIEW

Image processing techniques involve all the techniques that are used to enhance the graphic appearance of a given image or extract specific features from an image for better interpretation by human or automated system. Image processing, especially filtering has become important because there are many factors that affect the quality and purity of the images. For example, the change in pixel density or illumination will negatively affect the images; therefore noise should be removed using filtering. Image filtering is useful for many applications including smoothing, sharping, removing noise and edge enhancement. This project analyzes the various algorithms used for image filtering which improves image quality and clarity. The first section introduces different algorithms that are used in image filtering and how each algorithm processes the image. Then, a discussion and analysis of the project has been done to classify and compare between the algorithms studied.

## STRATEGIES

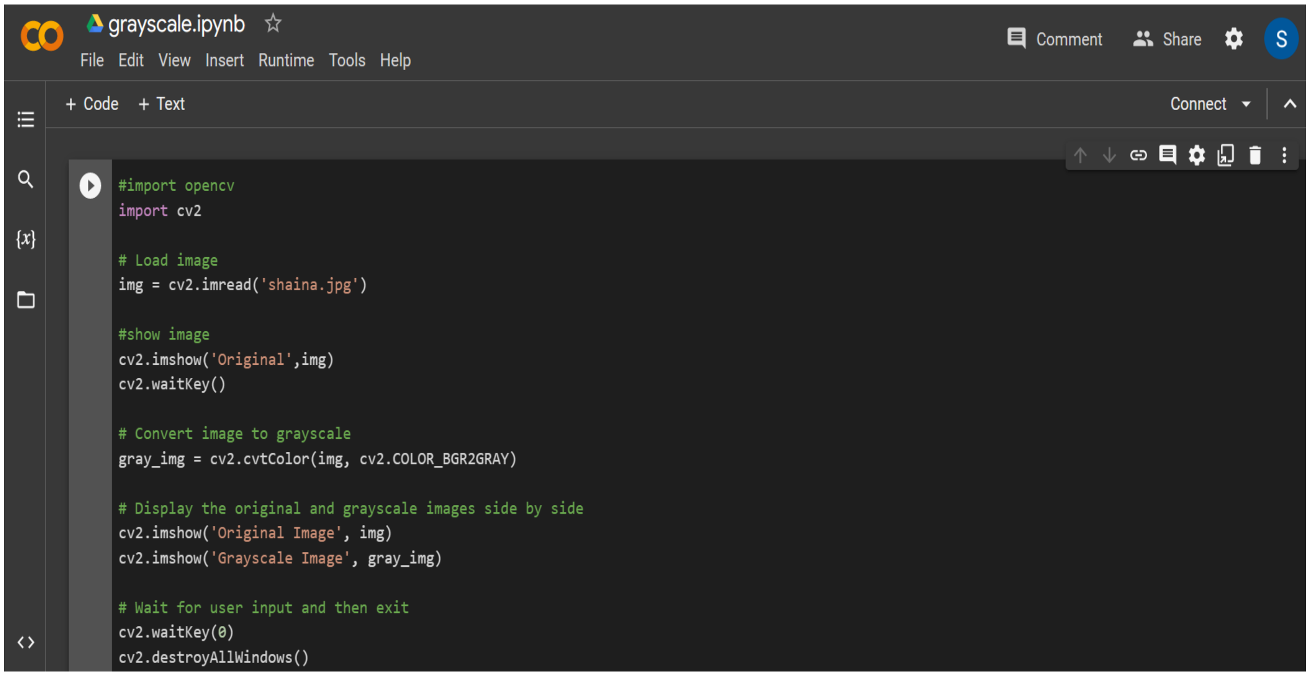
- Firstly , we installed pip , open cv , numpy , keras and deepface.
- We executed some of the algorithms through notepad . Image and code should be in the same directory . Then with the help of powershell we executed the code .
- Some of the algorithms were executed in a python extension of VS CODE that was launched through anaconda navigator .

## ALGORITHMS , CODES AND DEMONSTRATION OF IMAGE FILTERS

## ALGORITHM FOR GRAYSCALE

- Import the OpenCV library.
- Load the image named "shaina.jpg" using the cv2.imread() function and store it in a variable named "img".
- Show the original image using the cv2.imshow() function and wait for user input using the cv2.waitKey() function.
- Convert the original image to grayscale using the cv2.cvtColor() function and store it in a variable named "gray_img".
- Show both the original and grayscale images side by side using the cv2.imshow() function.
- Wait for user input using the cv2.waitKey() function.
- Close all windows using the cv2.destroyAllWindows() function

## CODE

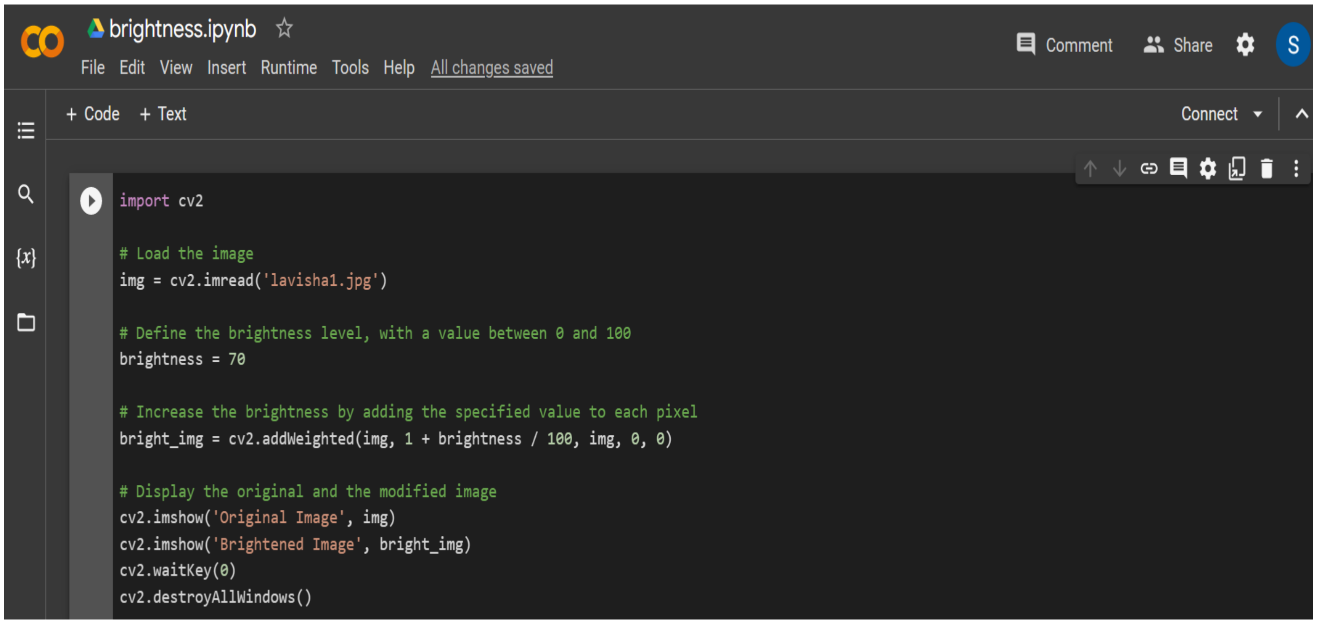


```
#import opencv
import cv2

# Load image
img = cv2.imread('shaina.jpg')

#show image
cv2.imshow('Original',img)
cv2.waitKey()

# Convert image to grayscale
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Display the original and grayscale images side by side
cv2.imshow('Original Image', img)
cv2.imshow('Grayscale Image', gray_img)

# Wait for user input and then exit
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## ALGORITHM FOR BRIGHTNESS

- Import the OpenCV library.
- Load the input image 'lavisha1.jpg' using the cv2.imread() function, and store it in the img variable.
- Define the desired brightness level using the brightness variable, which should be a value between 0 and 100.
- Calculate the scaling factor for brightness adjustment by adding 1 to the brightness level divided by 100.
- Use the cv2.addWeighted() function to blend the original image with itself, applying the calculated scaling factor as the weight for the source image.
- Store the resulting image in the bright_img variable.
- Display the original image and the brightened image side by side using the cv2.imshow() function, with window names 'Original Image' and 'Brightened Image' respectively.
- Use the cv2.waitKey() function to wait for a key event (e.g., pressing a key) before proceeding to the next step.
- Use the cv2.destroyAllWindows() function to close all open windows after a key event occurs, effectively terminating the program.

## CODE

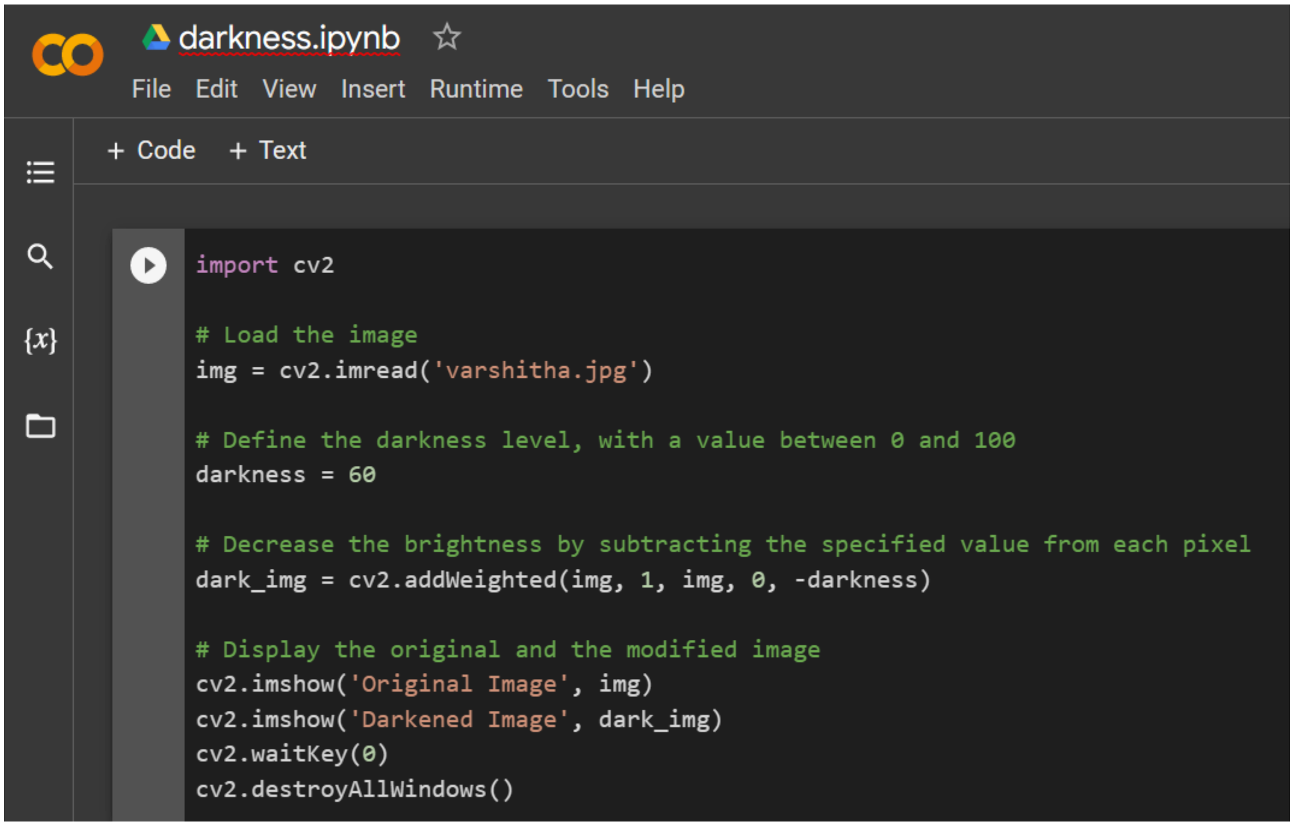


```python
import cv2

# Load the image
img = cv2.imread('lavisha1.jpg')

# Define the brightness level, with a value between 0 and 100
brightness = 70

# Increase the brightness by adding the specified value to each pixel
bright_img = cv2.addWeighted(img, 1 + brightness / 100, img, 0, 0)

# Display the original and the modified image
cv2.imshow('Original Image', img)
cv2.imshow('Brightened Image', bright_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## ALGORITHM FOR DARKNESS

- Import the OpenCV library.
- Load the image named "varshitha.jpg" using the cv2.imread() function.
- Define the darkness level as a variable named "darkness" , with a value between 0 and 100. This value determines the amount of brightness reduction.
- Use the cv2.addWeighted() function to decrease the brightness of each pixel in the image.
- This function takes the following arguments: a. img: the input image. b. alpha: weight of the first input image (in this case, it is 1). c. beta: weight of the second input image (in this case, it is 0). d. gamma: scalar added to each sum (in this case, it is -darkness).
- Store the darkened image in a variable named "dark_img".
- Display both the original and the darkened image using the cv2.imshow() function.
- Wait for a key press using the cv2.waitKey() function.
- Destroy all windows using the cv2.destroyAllWindows() function.

## CODE

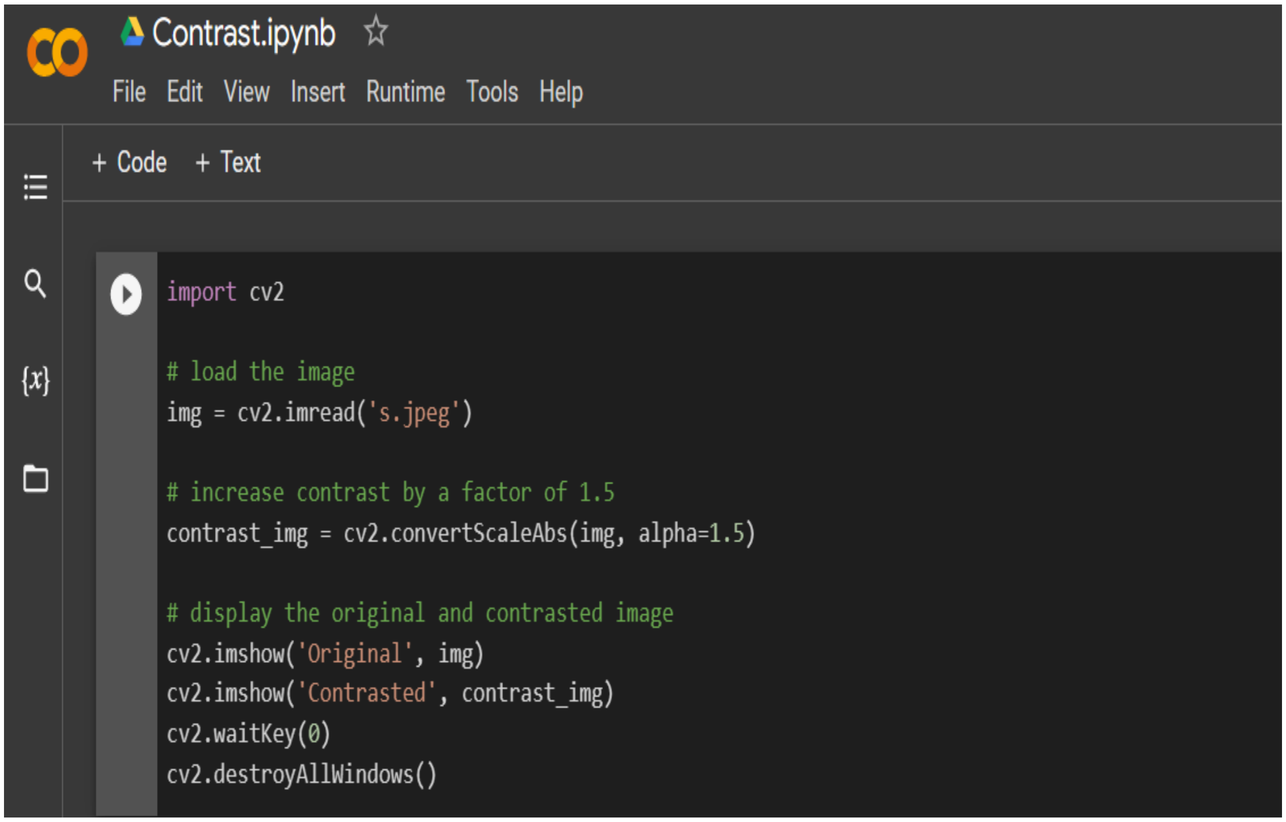


```python
import cv2

# Load the image
img = cv2.imread('varshitha.jpg')

# Define the darkness level, with a value between 0 and 100
darkness = 60

# Decrease the brightness by subtracting the specified value from each pixel
dark_img = cv2.addWeighted(img, 1, img, 0, -darkness)

# Display the original and the modified image
cv2.imshow('Original Image', img)
cv2.imshow('Darkened Image', dark_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## ALGORITHM FOR CONTRAST

- Import the OpenCV library using the "import cv2" statement.
- Load the image file using "cv2.imread()" method, and store the result in the "img" variable.
- Apply contrast adjustment using "cv2.convertScaleAbs()" method with an "alpha" value of 1.5, and store the result in the "contrast_img" variable.
- Display the original image and the contrasted image using the "cv2.imshow()" method.
- Pass "Original" and "Contrasted" as window names, and "img" and "contrast_img" as images to be displayed, respectively.
- Wait for a key press using the "cv2.waitKey(0)" method.
- This will wait indefinitely until a key is pressed.
- Destroy all open windows using the "cv2.destroyAllWindows()" method.

## CODE

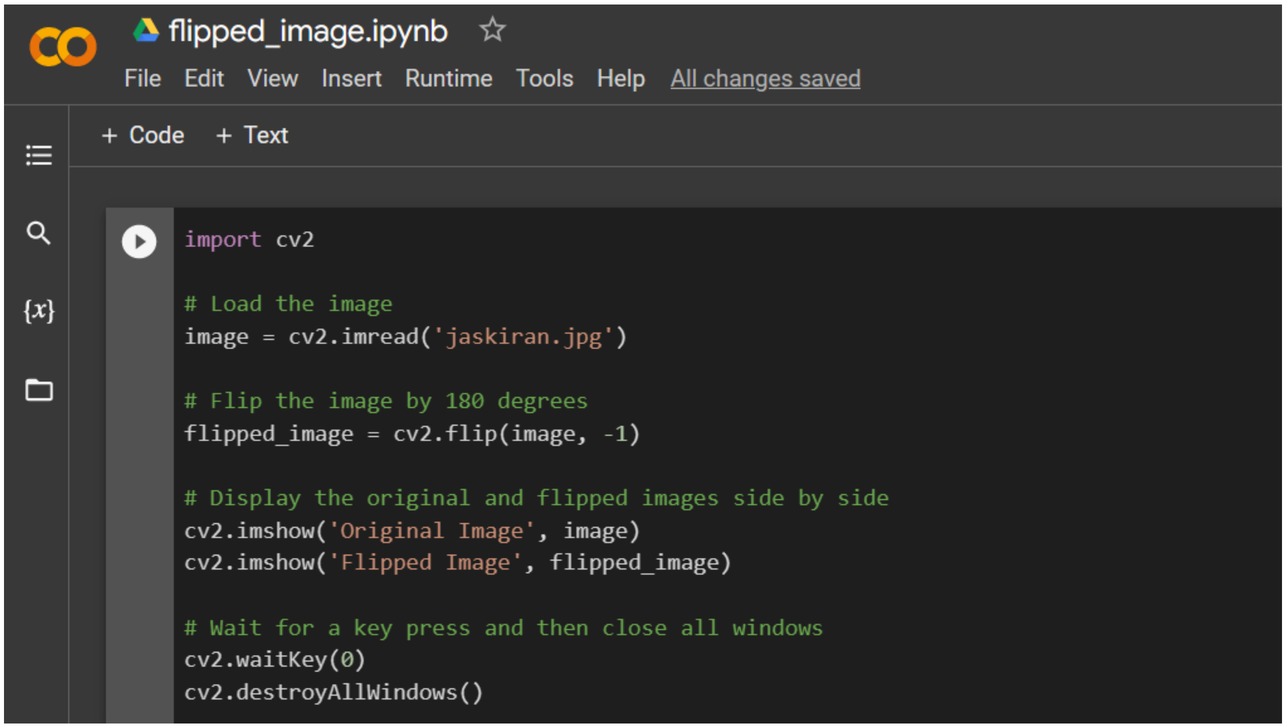


```python
import cv2

# load the image
img = cv2.imread('s.jpeg')

# increase contrast by a factor of 1.5
contrast_img = cv2.convertScaleAbs(img, alpha=1.5)

# display the original and contrasted image
cv2.imshow('Original', img)
cv2.imshow('Contrasted', contrast_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## ALGORITHM FOR FLIPPED IMAGE

- Import the OpenCV library. Load the image named "jaskiran.jpg" using the cv2.imread() function.
- Flip the image by 180 degrees using the cv2.flip() function with the following arguments: a. image: the input image. b. flipCode: the code that determines the direction of flipping.
- A value of 0 flips the image vertically, a value of 1 flips it horizontally, and a value of -1 flips it both vertically and horizontally.
- Store the flipped image in a variable named "flipped_image".
- Display both the original and flipped images side by side using the cv2.imshow() function.
- Wait for a key press using the cv2.waitKey() function.
- Close all windows using the cv2.destroyAllWindows() function.

## CODE

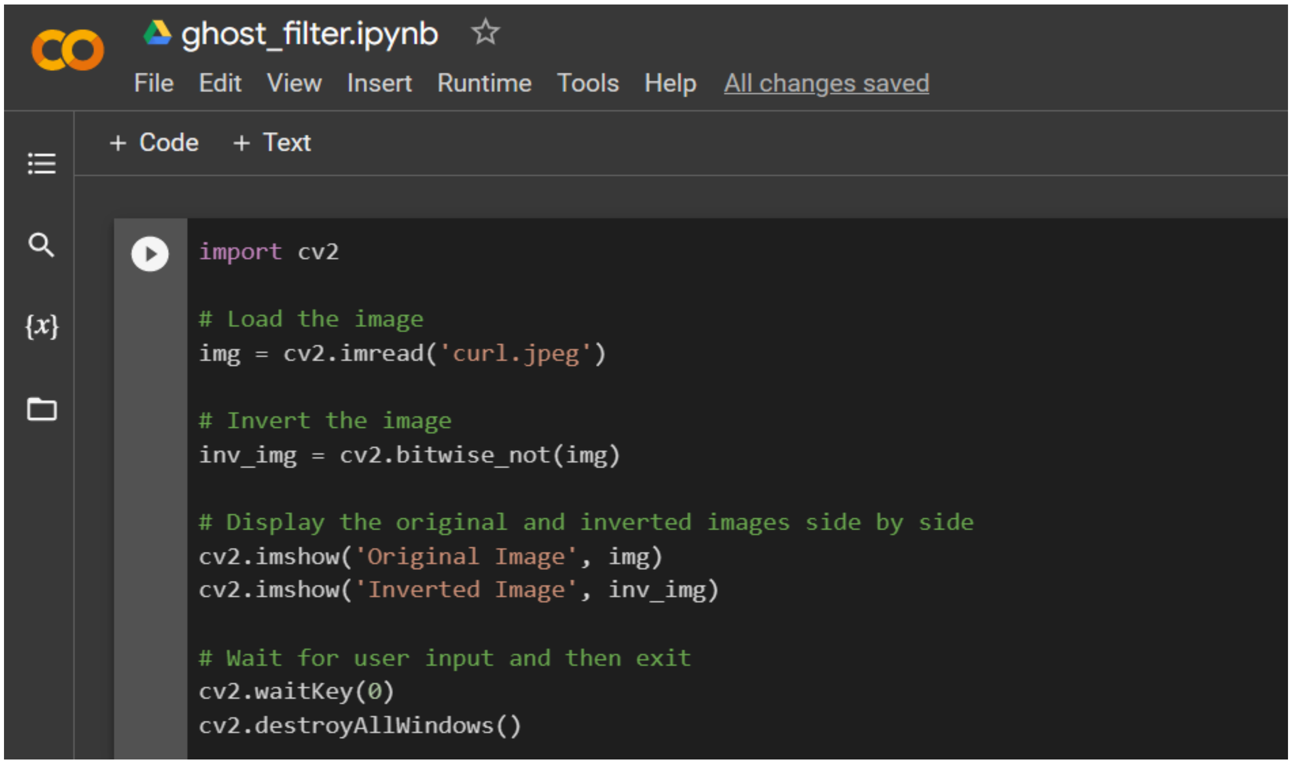


```python
import cv2

# Load the image
image = cv2.imread('jaskiran.jpg')

# Flip the image by 180 degrees
flipped_image = cv2.flip(image, -1)

# Display the original and flipped images side by side
cv2.imshow('Original Image', image)
cv2.imshow('Flipped Image', flipped_image)

# Wait for a key press and then close all windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## ALGORITHM FOR GHOST FILTER

- Import the OpenCV library. Load the image named "curl.jpeg" using the cv2.imread() function.
- Invert the image using the cv2.bitwise_not() function, which computes the bitwise NOT operation on each pixel of the input image.
- Store the inverted image in a variable named "inv_img".
- Display both the original and inverted images side by side using the cv2.imshow() function.
- Wait for user input using the cv2.waitKey() function.
- Close all windows using the cv2.destroyAllWindows() function.

## CODE

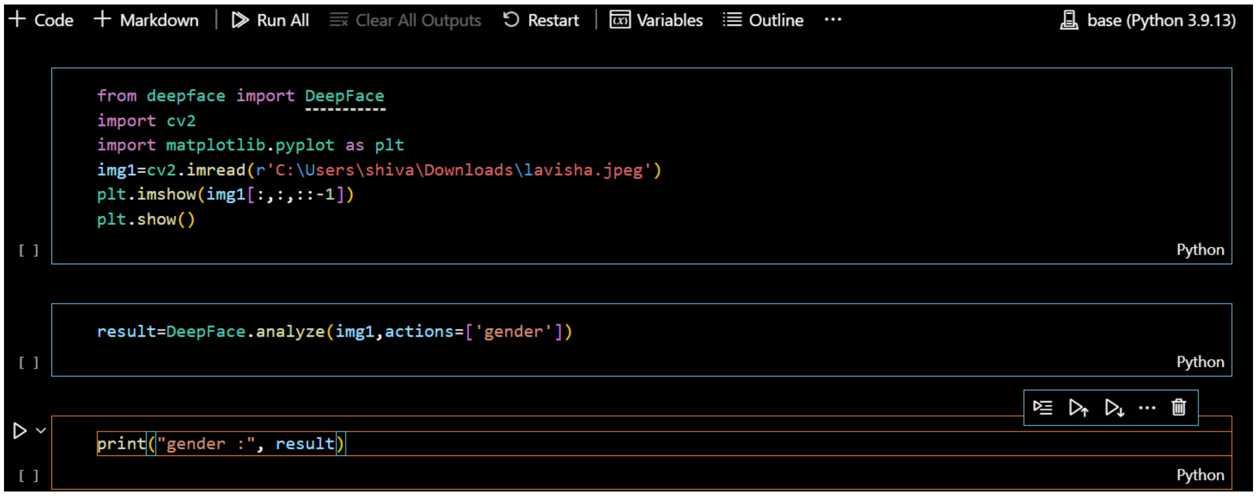


```python
import cv2

# Load the image
img = cv2.imread('curl.jpeg')

# Invert the image
inv_img = cv2.bitwise_not(img)

# Display the original and inverted images side by side
cv2.imshow('Original Image', img)
cv2.imshow('Inverted Image', inv_img)

# Wait for user input and then exit
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## ALGORITHM FOR GENDER DETECTION

- Import the necessary libraries: deepface library for analyzing face attributes cv2 for reading and manipulating images matplotlib for displaying the image Load the image using cv2.imread() function and store it in a variable img1
- Display the image using plt.imshow() function and plt.show() function.
- Analyze the gender of the image using the DeepFace.analyze() function with actions parameter as ['gender'] and store the result in a variable result.
- Print the gender result using print() function

## CODE



```python
from deepface import DeepFace
import cv2
import matplotlib.pyplot as plt
img1=cv2.imread(r'C:\Users\shiva\Downloads\lavisha.jpeg')
plt.imshow(img1[:,:,::-1])
plt.show()
```

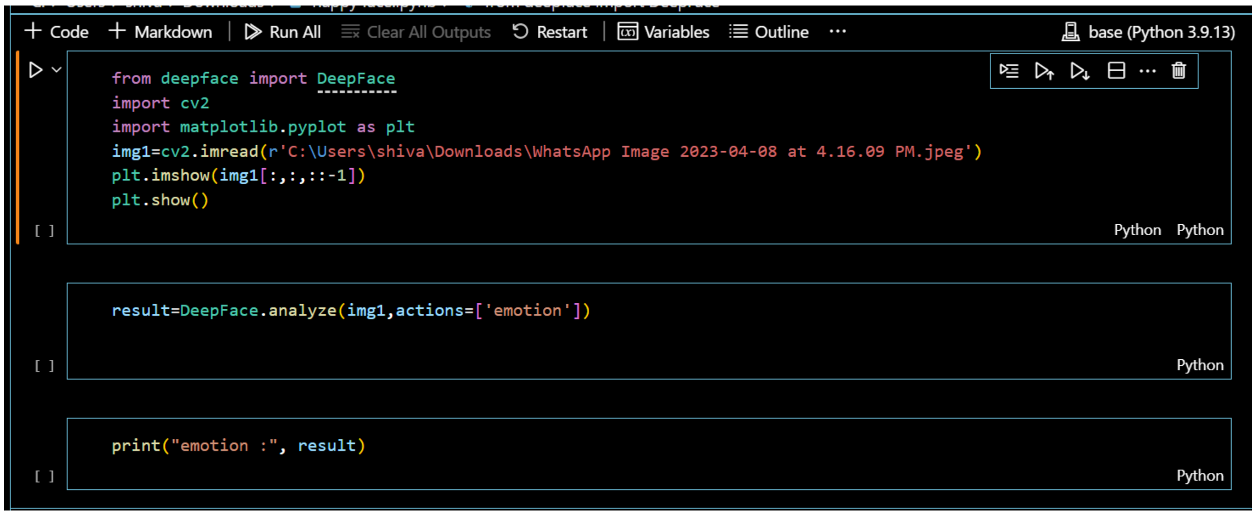
```python
result=DeepFace.analyze(img1,actions=['gender'])
```

```python
print("gender :", result)
```

## ALGORITHM FOR EMOTION DETECTION

- Import necessary libraries: from deepface import DeepFace import cv2 import matplotlib.pyplot as plt Load the image using cv2.imread() function: img1=cv2.imread(r'')
- Display the loaded image using matplotlib.pyplot.imshow() function: plt.imshow(img1[:,:,::-1]) Show the image using matplotlib.pyplot.show() function: plt.show()
- Analyze the loaded image using DeepFace.analyze() function with the emotion action specified: result=DeepFace.analyze(img1,actions=['emotion'])
- Display the emotion result using print() function: print("emotion :" , result)

## CODE



```
from deepface import DeepFace
import cv2
import matplotlib.pyplot as plt
img1=cv2.imread(r'C:\Users\shiva\Downloads\WhatsApp Image 2023-04-08 at 4.16.09 PM.jpeg')
plt.imshow(img1[:,:,::-1])
plt.show()
```

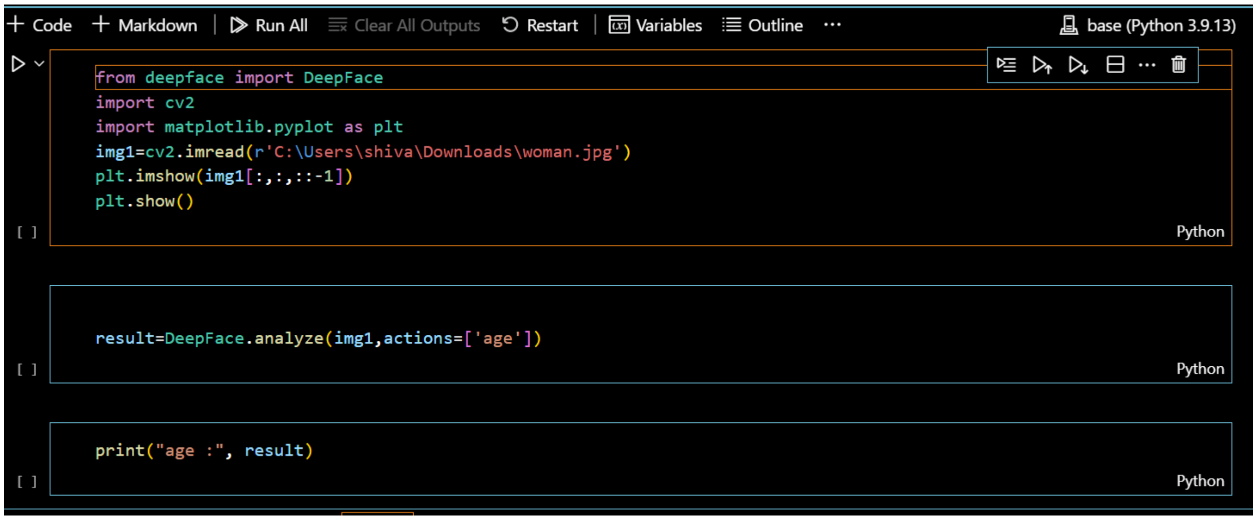
```
result=DeepFace.analyze(img1,actions=['emotion'])
```

```
print("emotion :", result)
```

## ALGORITHM FOR AGE DETECTION

- import deepface, cv2, matplotlib.pyplot
- img1 = cv2.imread(r'C:\Users\shiva\Downloads\WhatsApp Image 2023-04-08 at 4.16.09 PM.jpeg')
- plt.imshow(img1[:,:,::-1])
- plt.show()
- result = DeepFace.analyze(img1, actions=['age']) 6. print("age :" , result)

## CODE



```
from deepface import DeepFace
import cv2
import matplotlib.pyplot as plt
img1=cv2.imread(r'C:\Users\shiva\Downloads\woman.jpg')
plt.imshow(img1[:,:,::-1])
plt.show()
```

```
result=DeepFace.analyze(img1,actions=['age'])
```

```
print("age :", result)
```

## APPLICATIONS

- IMAGE ENHANCEMENT:This is commonly used in photography, video editing to improve the visibility of important features and details in images.

- MEDICAL IMAGING: These techniques are crucial for analyzing medical images such as X-rays, CT scans, and MRI scans for diagnosis and treatment planning.

- COMPUTER VISION: Computer vision applications include object recognition, face detection, image recognition, and scene understanding, which are used in fields such as autonomous vehicles, surveillance systems, and robotics.

- IMAGE RECOGNITION: This is used in applications such as image-based search engines, content-based image retrieval, and image-based recommendation systems.

- AUGMENTED REALITY (AR) : This is used in areas such as gaming, entertainment, training, and simulation.

- SECURITY: These techniques are used in applications such as video surveillance, access control, and biometric identification.

- INDUSTRIAL AUTOMATION: These techniques are used in industries such as automotive, electronics, pharmaceuticals, and food processing to ensure product quality and reduce manufacturing defects.

- SOCIAL MEDIA AND ENTERTAINMENT: Used in applications for tasks such as image editing, special effects, and image-based filters and overlays for photos and videos on social media platforms, photo sharing apps.

## FUTURE SCOPE

- Advanced Image Enhancement: This includes techniques such as deep learning based image superresolution, denoising, deblurring, and other advanced image enhancement algorithms that can restore or enhance images with higher fidelity and accuracy.

- Real-time Image Processing : Future developments may include more efficient and optimized algorithms for real time image processing, as well as hardware acceleration techniques to enable real time image processing on resource constrained devices.

- Computational Photography: Future advancements in computational photography may include technologies such as light field imaging, plenoptic imaging, multi-camera systems, and other novel imaging techniques that can enable new types of visual experiences and imaging capabilities.

- Image Forensics and Security: reliable techniques for image forensics and security, including deep learning-based approaches for detecting sophisticated image manipulations and developing encryption techniques for secure image transmission and storage.

- Image-based Human-Computer Interaction: Future developments include more advanced algorithms for accurate and natural interaction, as well as integration with other emerging technologies such as virtual reality, augmented reality .

- Mobile and Edge Computing : This includes techniques such as cloud-assisted image processing, on-device image processing, and distributed image processing to enable more powerful and responsive image processing capabilities on mobile and edge devices.