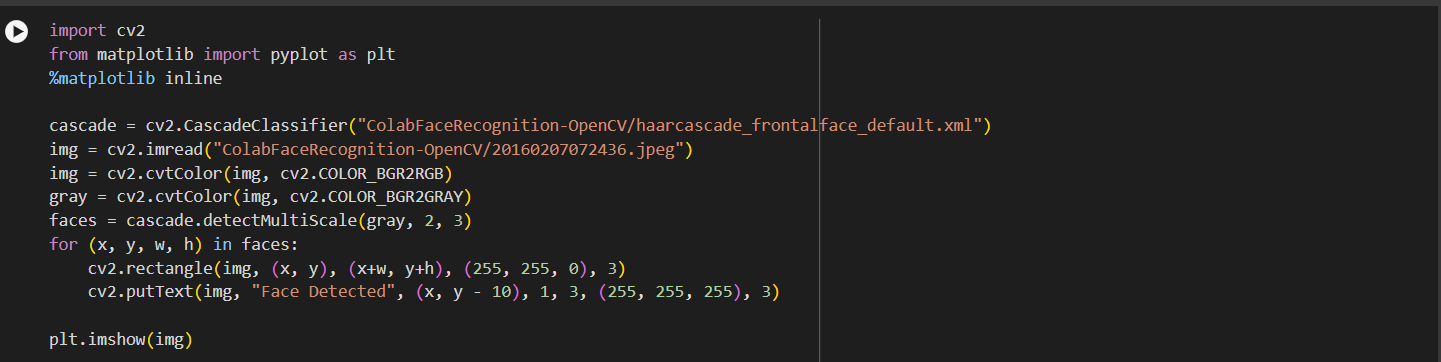
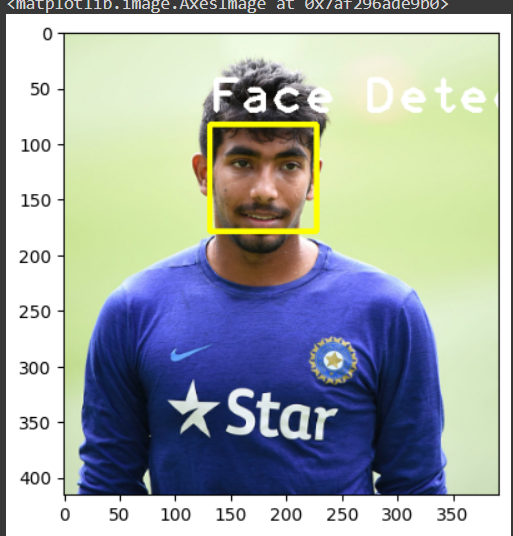
FACE DETECTION PROJECT

USING OPENCV

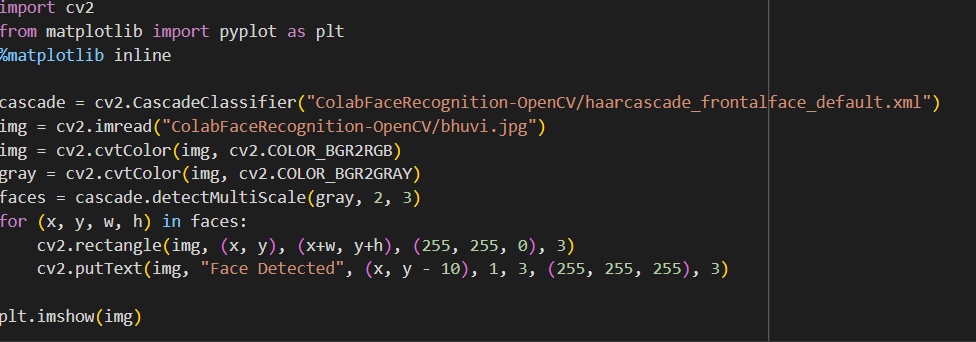
Now we are going to see the program of face detection.



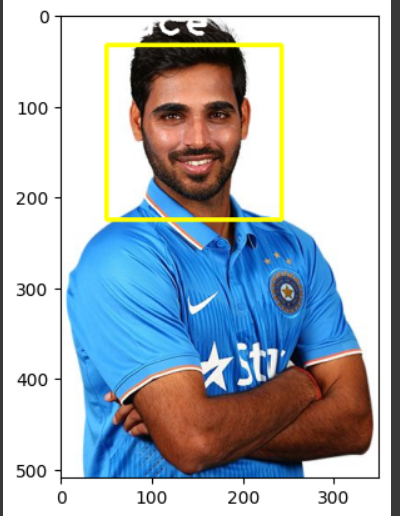
OUTPUT:



Another Example:



OUTPUT:



EXPLANATION OF ABOVE PROGRAM:

**Step-by-Step Explanation**

1. **Import Necessary Libraries:** You import the necessary libraries: OpenCV for image processing and Matplotlib for displaying the image.

python

Copy code

import cv2

from matplotlib import pyplot as plt

%matplotlib inline

1. **Load Haar Cascade Classifier:** You load a pre-trained Haar cascade classifier for frontal face detection.

python

Copy code

cascade = cv2.CascadeClassifier("ColabFaceRecognition-OpenCV/haarcascade\_frontalface\_default.xml")

1. **Read and Preprocess the Image:** You read an image, convert it from BGR to RGB (as OpenCV reads images in BGR by default), and then convert it to grayscale for the face detection algorithm.

python

Copy code

img = cv2.imread("ColabFaceRecognition-OpenCV/bhuvi.jpg")

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

1. **Detect Faces:** You use the detectMultiScale method to detect faces in the grayscale image. The parameters used here are:
   * gray: The input image in grayscale.
   * scaleFactor: Specifies how much the image size is reduced at each image scale. You used 2.
   * minNeighbors: Specifies how many neighbors each candidate rectangle should have to retain it. You used 3.

python

Copy code

faces = cascade.detectMultiScale(gray, 2, 3)

1. **Draw Rectangles and Labels:** For each detected face, you draw a rectangle around it and add a text label.

python

Copy code

for (x, y, w, h) in faces:

cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 3)

cv2.putText(img, "Face Detected", (x, y - 10), 1, 3, (255, 255, 255), 3)

1. **Display the Image:** Finally, you use Matplotlib to display the image with detected faces.

python

Copy code

plt.imshow(img)

plt.axis('off')

plt.show()