**SOURCE CODE**

import cv2

from deepface import DeepFace

import os

import concurrent.futures

# Load the Haar Cascade for face detection

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

# Path to the image

start = os.path.join('C:', 'Users', 'user', 'Desktop', 'myjoy2.jpg')

# Function to analyze a face region

def analyze\_face(face\_img):

try:

analyze = DeepFace.analyze(face\_img, actions=['age', 'gender'], enforce\_detection=False)

age = analyze[0]['age'] if isinstance(analyze, list) else analyze['age']

gender = analyze[0]['gender'] if isinstance(analyze, list) else analyze['gender']

return age, gender

except Exception as e:

print(f"Error analyzing face: {e}")

return None, None

# Read the image

img = cv2.imread(start)

if img is None:

raise FileNotFoundError(f"Image file not found at {start}")

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

# Detect faces using Haar Cascade

faces = face\_cascade.detectMultiScale(grey, scaleFactor=1.1, minNeighbors=4, minSize=(30, 30))

# Analyze each detected face

with concurrent.futures.ThreadPoolExecutor() as executor:

future\_to\_face = {executor.submit(analyze\_face, img[y:y+h, x:x+w]): (x, y, w, h) for (x, y, w, h) in faces}

for future in concurrent.futures.as\_completed(future\_to\_face):

x, y, w, h = future\_to\_face[future]

age, gender = future.result()

if age is not None and gender is not None:

# Draw rectangle around the face

cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 5)

# Display age and gender on the image

label = f"Age: {age}, Gender: {gender}"

cv2.putText(img, label, (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 0, 255), 2)

# Save the result to a file

output\_path = os.path.join('C:', 'Users', 'user', 'Desktop', 'analyzed\_image.jpg')

cv2.imwrite(output\_path, img)

print(f"Processed image saved at {output\_path}")

# Display the image

cv2.imshow('img', img)

cv2.waitKey(0)

cv2.destroyAllWindows()



