MANUAL FOR STUDY & ANALYSIS OF HUMAN EMOTION

RAJARSHI RAYI, APARNA, HARIKA, NUNANDINI
RGUKT SRIKAKULAM

STUDY & ANALYSIS OF HUMAN EMOTIONS WHILE DRIVING

A project report submitted to

Rajiv Gandhi University of Knowledge Technologies

SRIKAKULAM

In partial fulfillment of the requirements for the Award of the degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

Submitted by
3rd year B. Tech 2nd Semester

R RAJARSHI (S171100)
J APARNA (S170837)
Y NUNANDINI (s171120)
R HARIKA(S170107)

Under the Esteemed Guidance of
Asst. Prof. Sri N Sesha Kumar sir

1 COURSE OBJECTIVE:

1. Able to learn How to detect the faces using Face Recognition library and using haarcascade file.
2. Learn about How to recognize the Face. For our datasets.
3. Learns How to detect the emotions of the faces.

2 WHAT WE LEARN:

1. How to install the Anaconda Navigator?
2. How to install the Python?
3. How to install the opencv, face_recognition, deepface?
4. How to get haarcascade file?
5. How to do Face detection from photos and Live demo mode?
6. How to recognize faces from given dataset via photos and live mode?
7. How to detect the emotion of a person via photos and Live mode?

3 INDEX

4 SYSTEM REQUIREMENTS:

- Anaconda Navigator (anaconda3)
- Opencv library
- Face Recognition library
- Deep face library
- Haarcascasde file (haarcascade_frontal_face.xml)

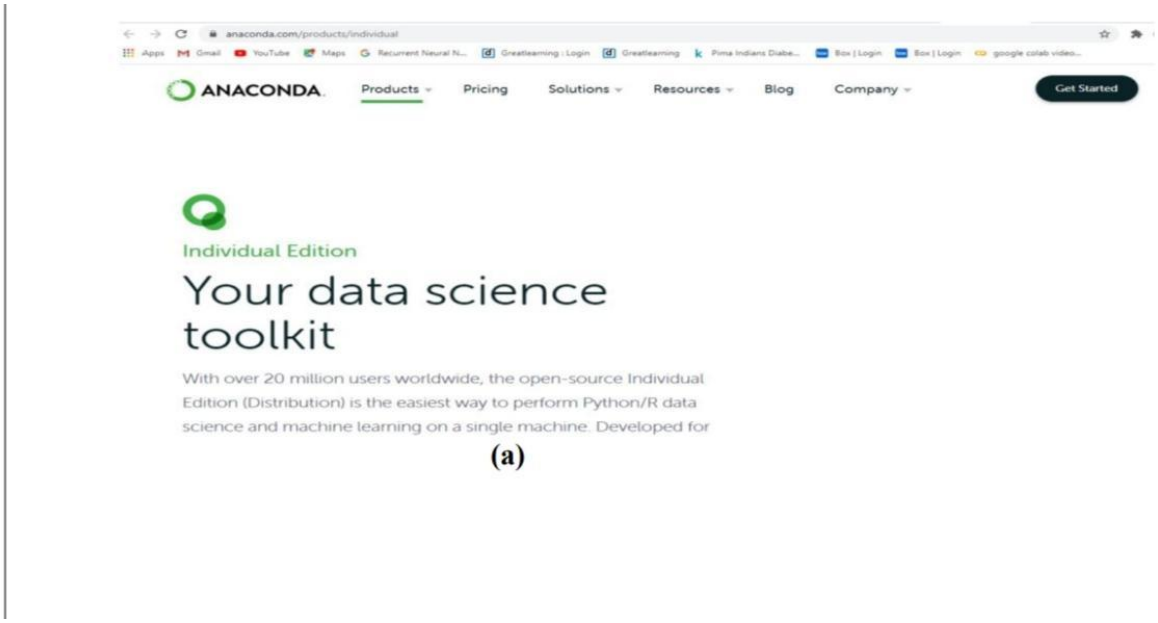
4.1 INSTALLATION OF ANACONDA NAVIGATOR:

Anaconda is an open-source platform and software that contains Jupyter, spider, etc. which can be used for processing large data, heavy scientific computing, and data analytics. Anaconda works for python and R programming languages. Jupyter Notebook is an open-source web application that enables us to create and share documents that contain live code, equations, visualizations, and narrative text. The uses of Jupyter notebook include data cleaning, data transformation, numerical simulation, statistical modeling, data visualization, machine learning, and many more. Jupyter notebook has support for over 40 different programming languages and Python is one of them. Python is needed (Python 3.3 or greater, or Python 2.7) for installing the Jupyter Notebook itself.

In order to install the Jupyter notebook using Anaconda, kindly follow the below instructions:

1. Open web Browser and search the anaconda download and click the first link

Then it shows this page.



(a)

2. Navigate to the Anaconda.com/downloads website

(b)

3. Choose a respective platform: Windows/Mac/Linux

(c)

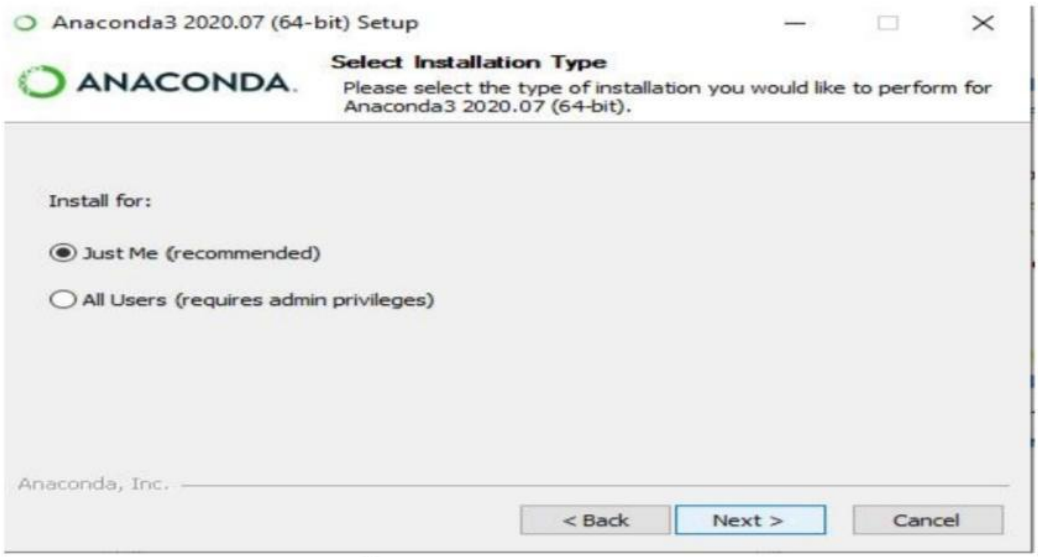
4. Click on and download the .exe installer

(d)

5. Now open and proceed to execute the .exe installer

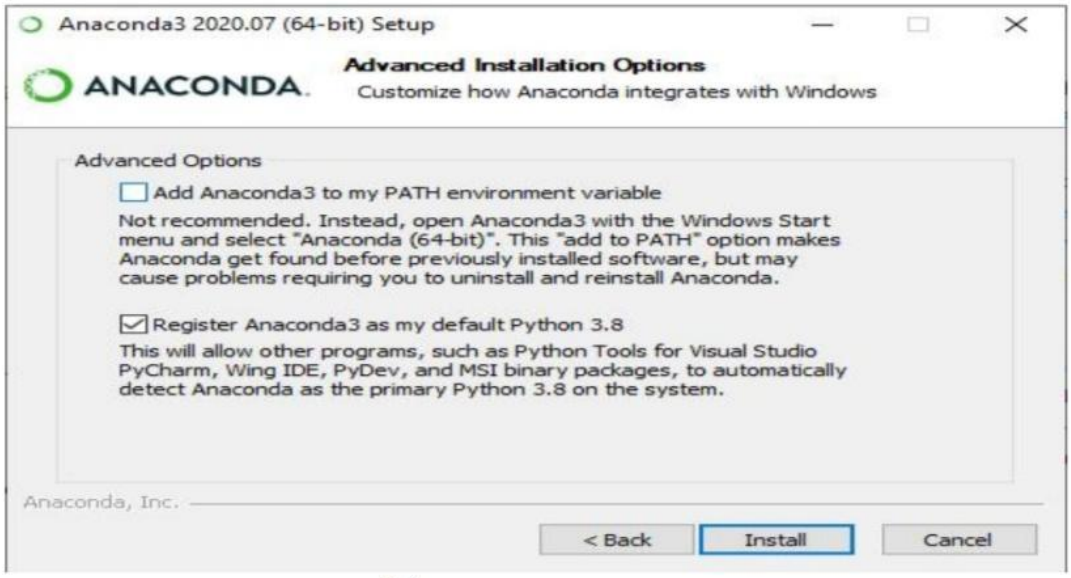
(e)

6. Proceed according to the steps and agree with the terms and services



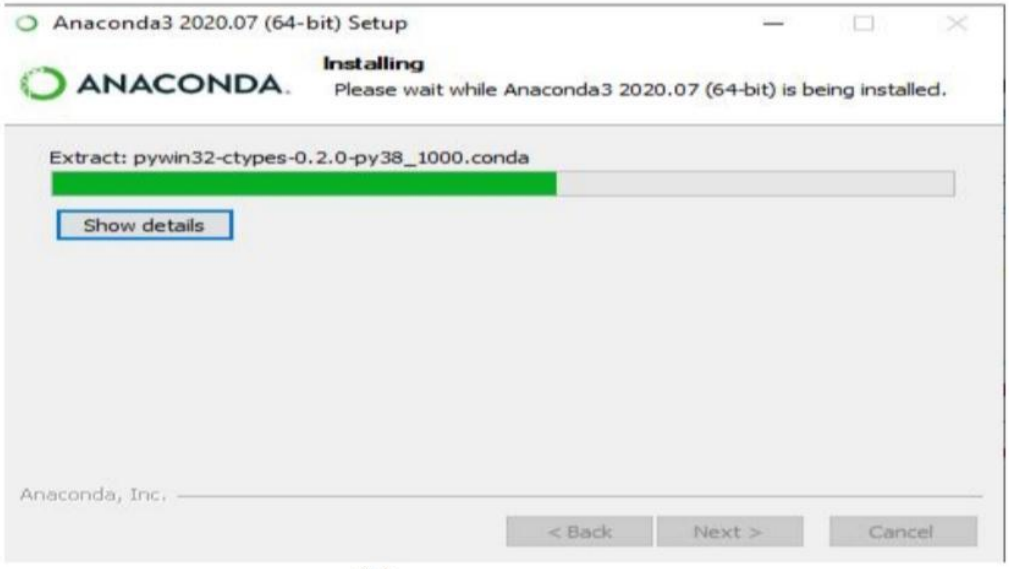
(f)

7. In order to start the installation process, click on install:



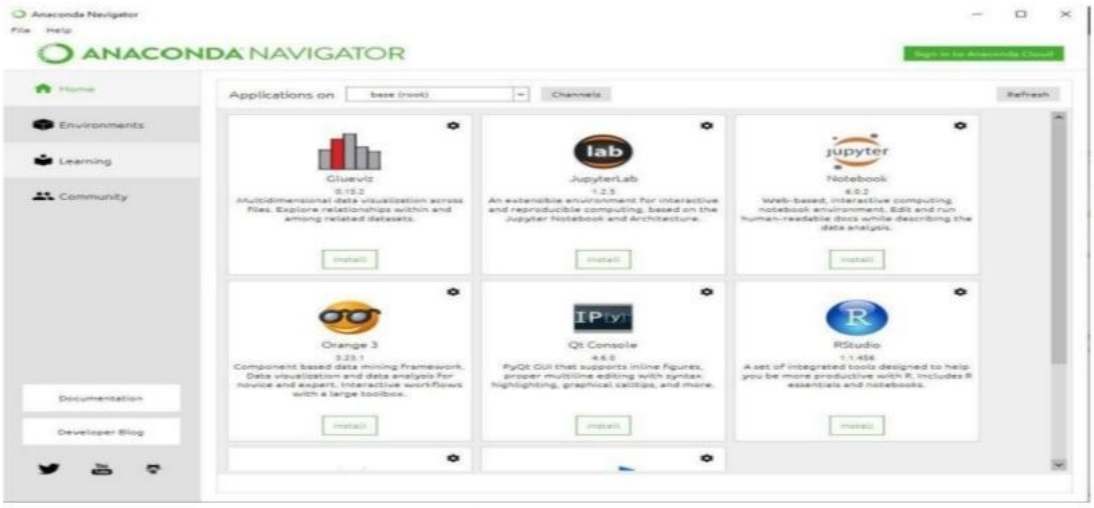
(g)

8. Installation begins:



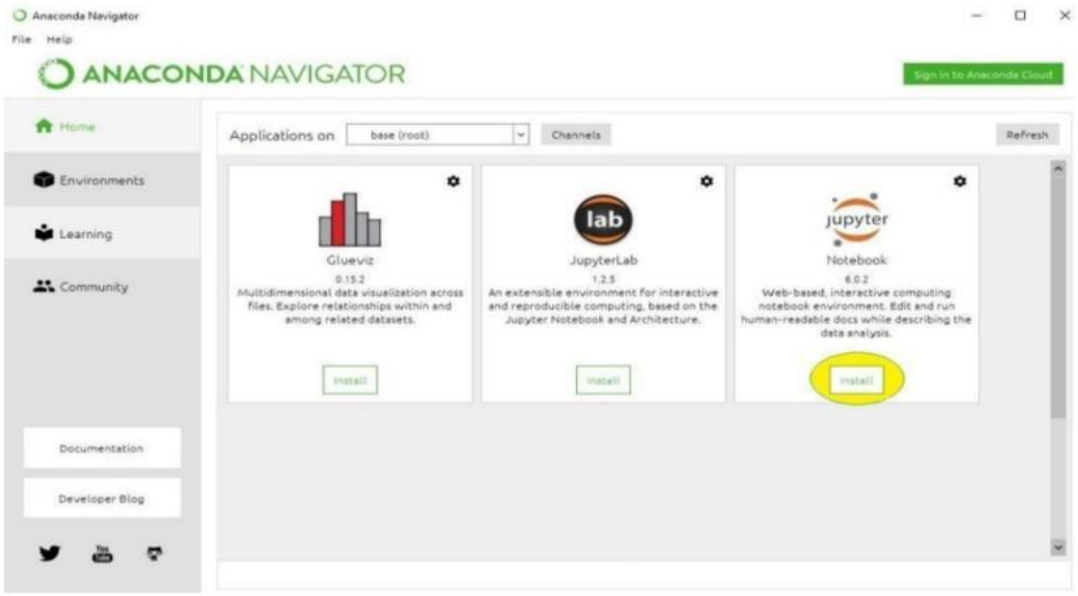
(h)

9. Installation of anaconda is finished
10. Now Launch the Anaconda Navigator



(i)

11. Click on the install Jupyter Notebook Option:

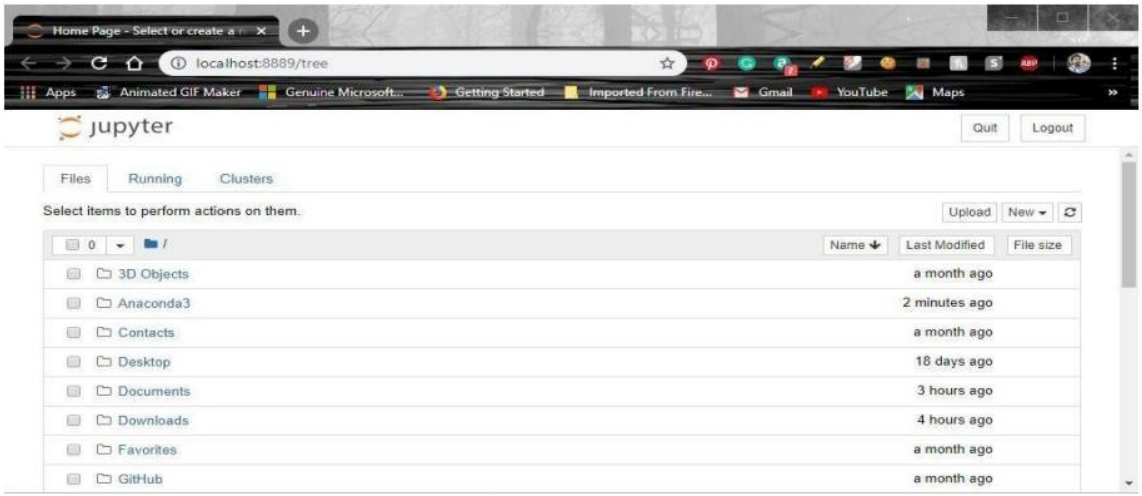


(j)

12. Opening the Jupyter Notebook

(k)

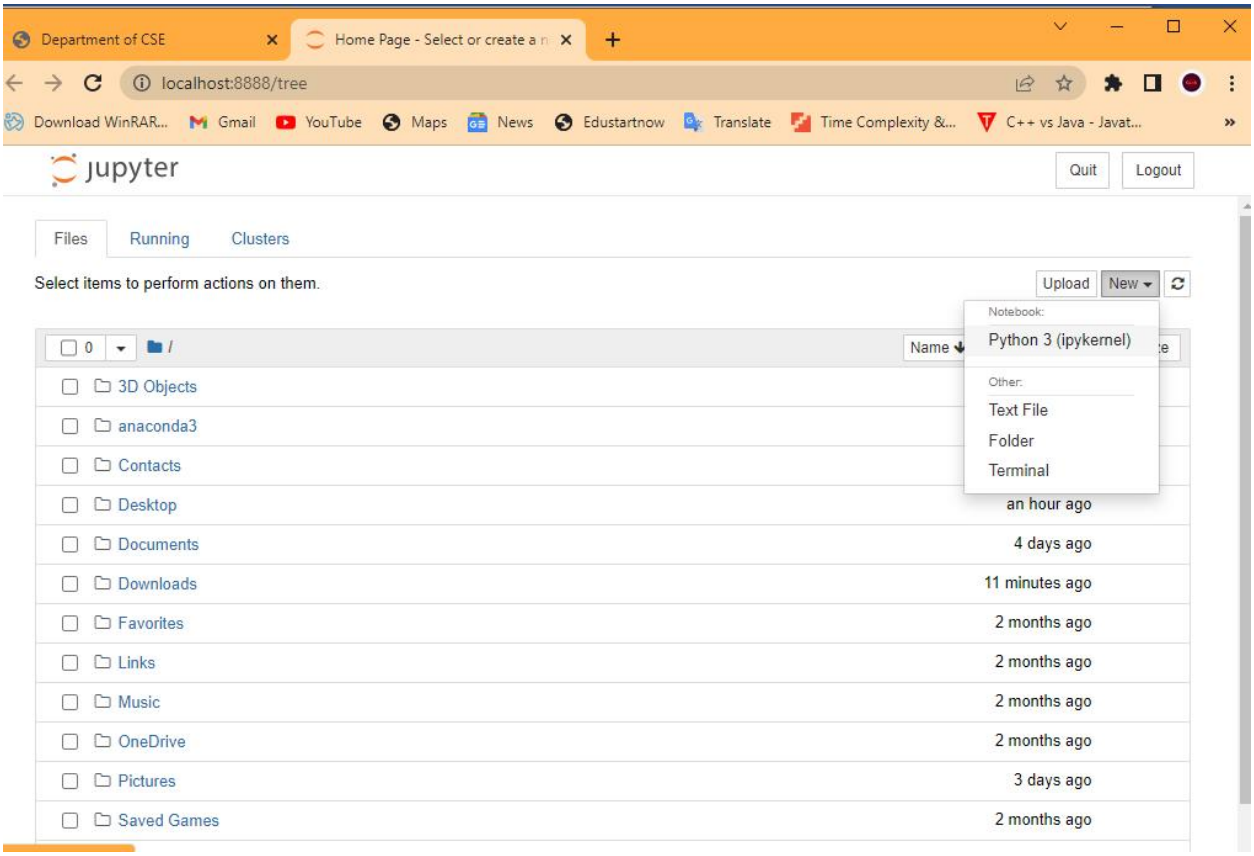
After successful installation clicks on Launch Jupyter Notebook



(m)

After opening of this page, you may open the python file using

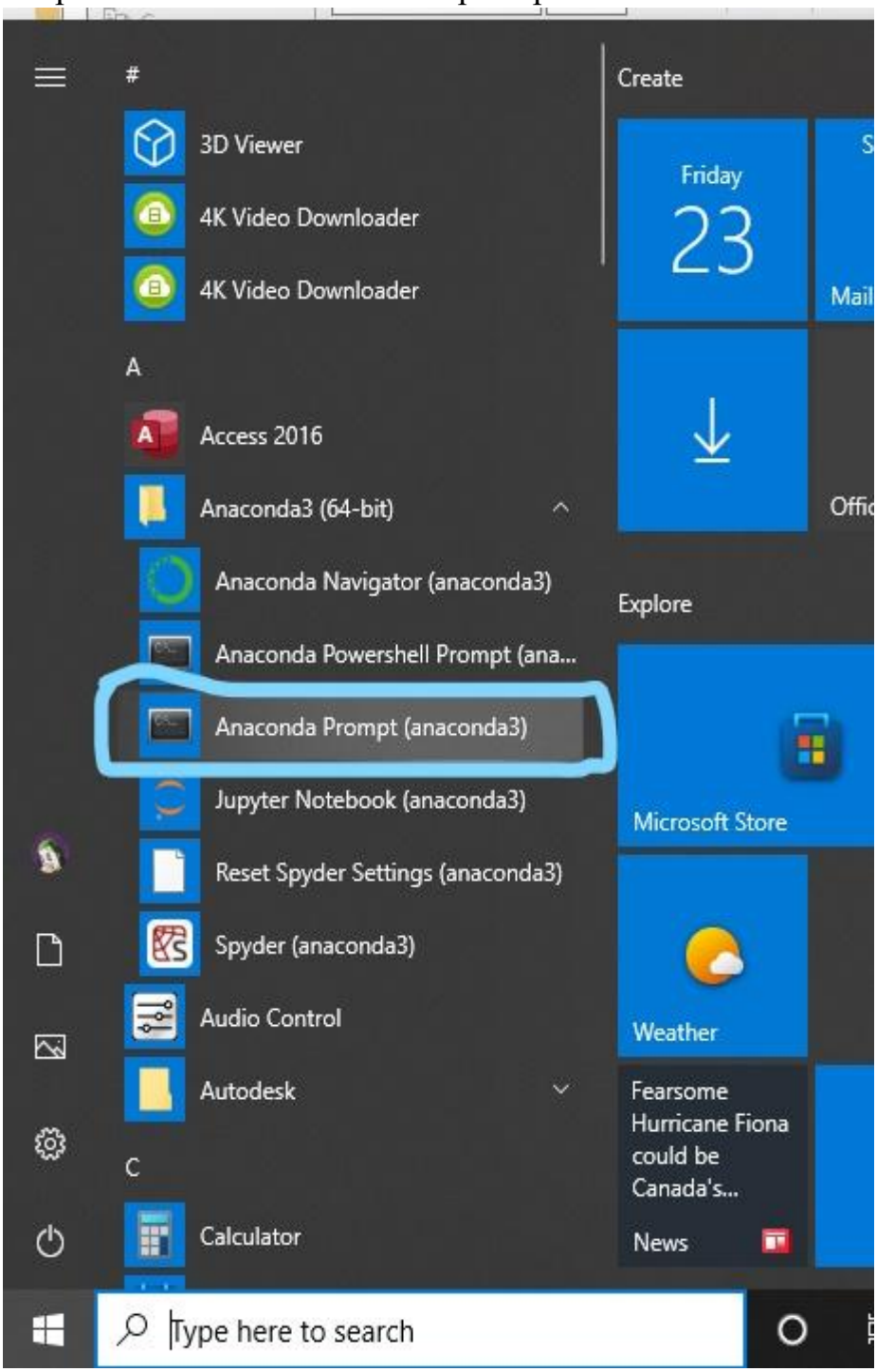
"New"→Python3



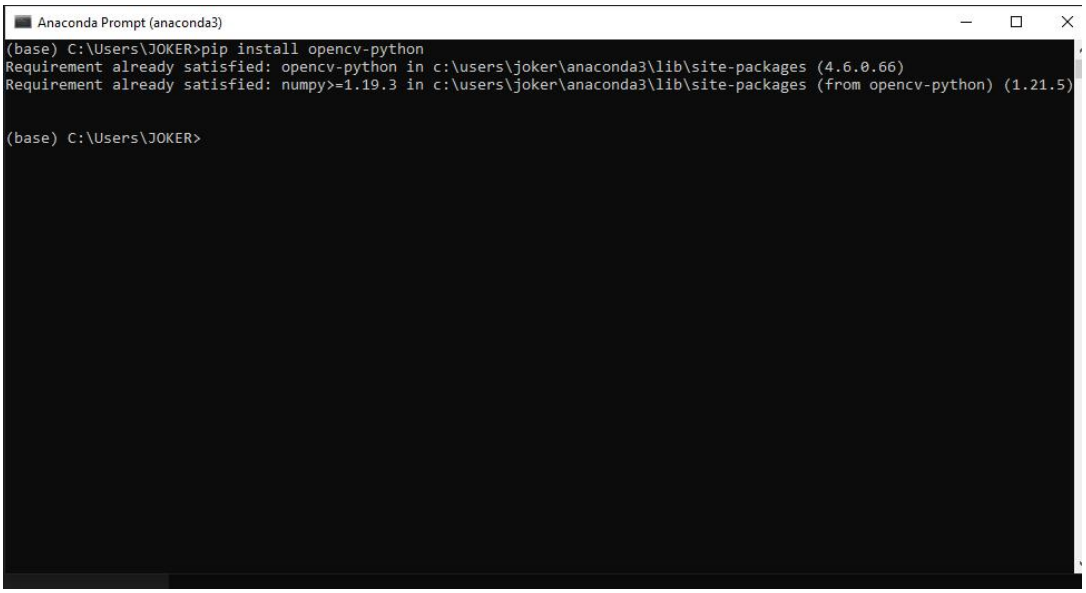
Congratulations You successfully Installed Anaconda Navigator.

4.2 INSTALLATION OF OPENCV LIBRARY:

1. Open the anaconda command prompt.

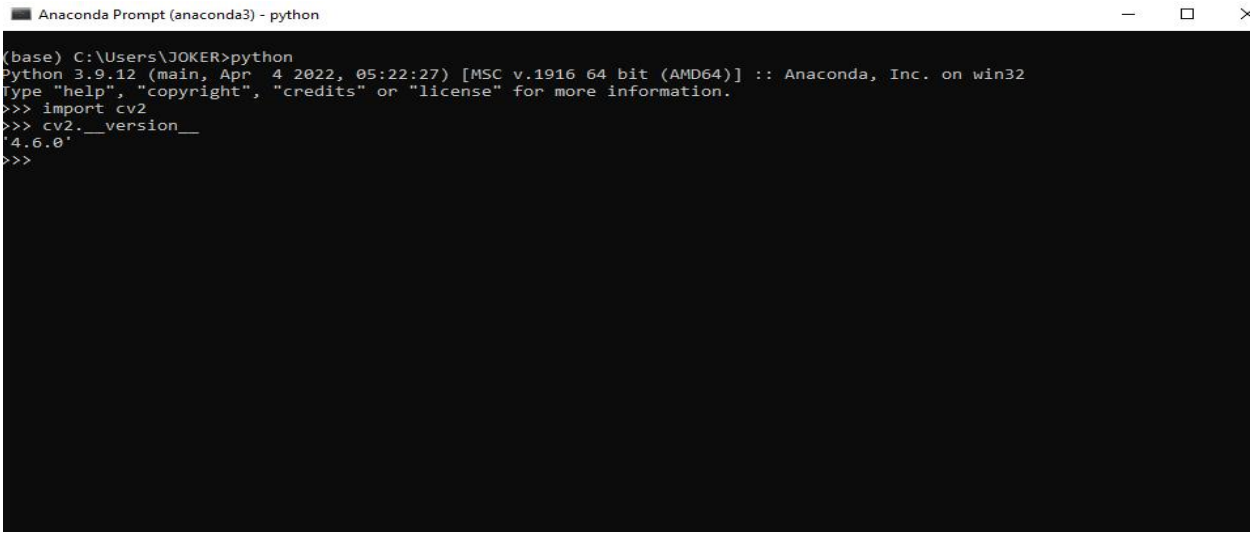


2. Now type command “pip install opencv-python”. Wait for some time for downloading and installing approximately 5 to 10 min.



* Here: I already installed so it displays this.

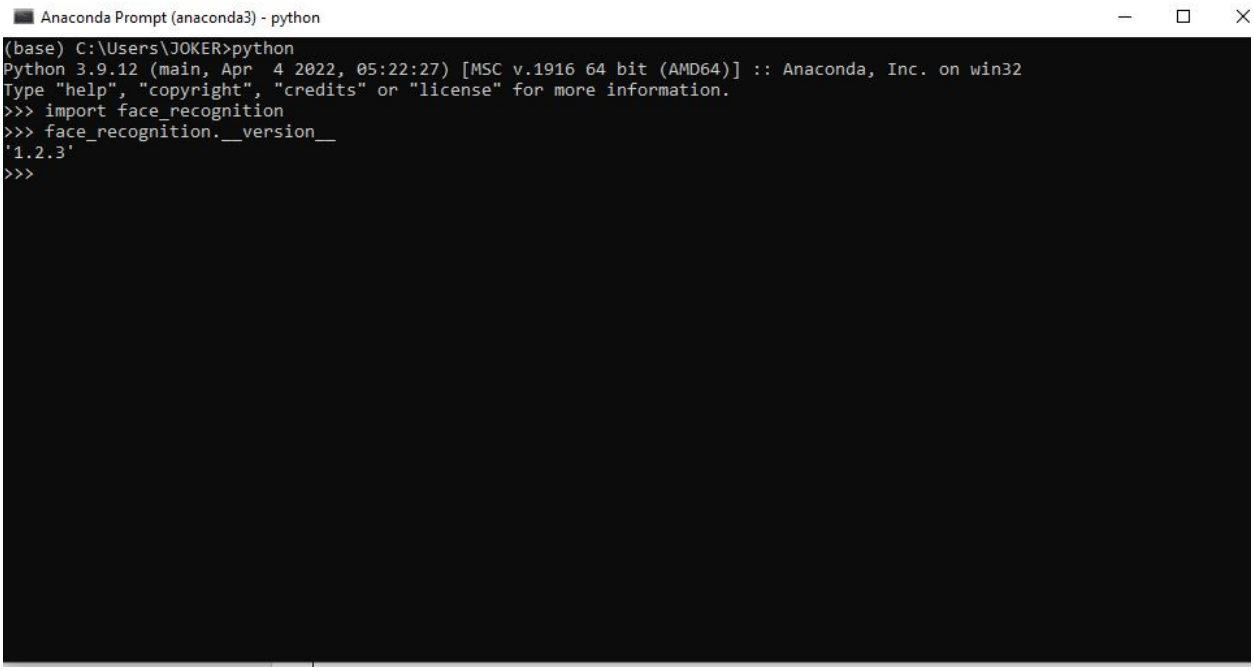
3. After the installation in same command prompt type “python”. Click enter.
4. Python interpreter opens then use command “import cv2”. If it executes without error means it is successfully installed.
    a. If Error means find appropriate solution using the error message and search in Internet.
    b. Major cause of error is cam access problem in the system.
5. After successful of execution then type command in python interpreter.”cv2. __version__”
    a. It shows the current version of opencv file that you installed try to use updated version.



Congratulations on successful installation of Opencv.

4.3 INSTALLATION OF FACE RECOGNITION:

1. After installation of opencv in same prompt use this command for face recognition "**conda install -c conda-forge face_recognition**"
2. If this face_recognition is done with this single command to verify that use same process as we did for opencv library.
3. Python→"import face_recognition"→"face_recognition. __version__"



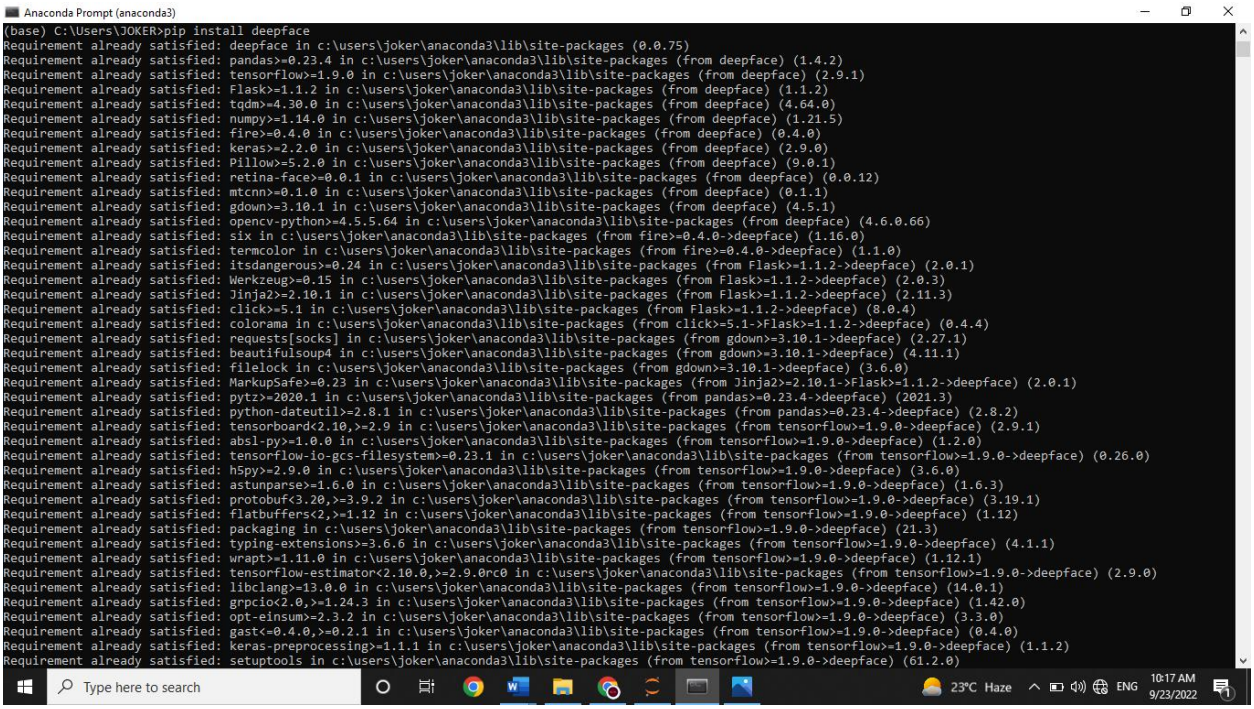
Congratulation for successful installation of Face recognition library.

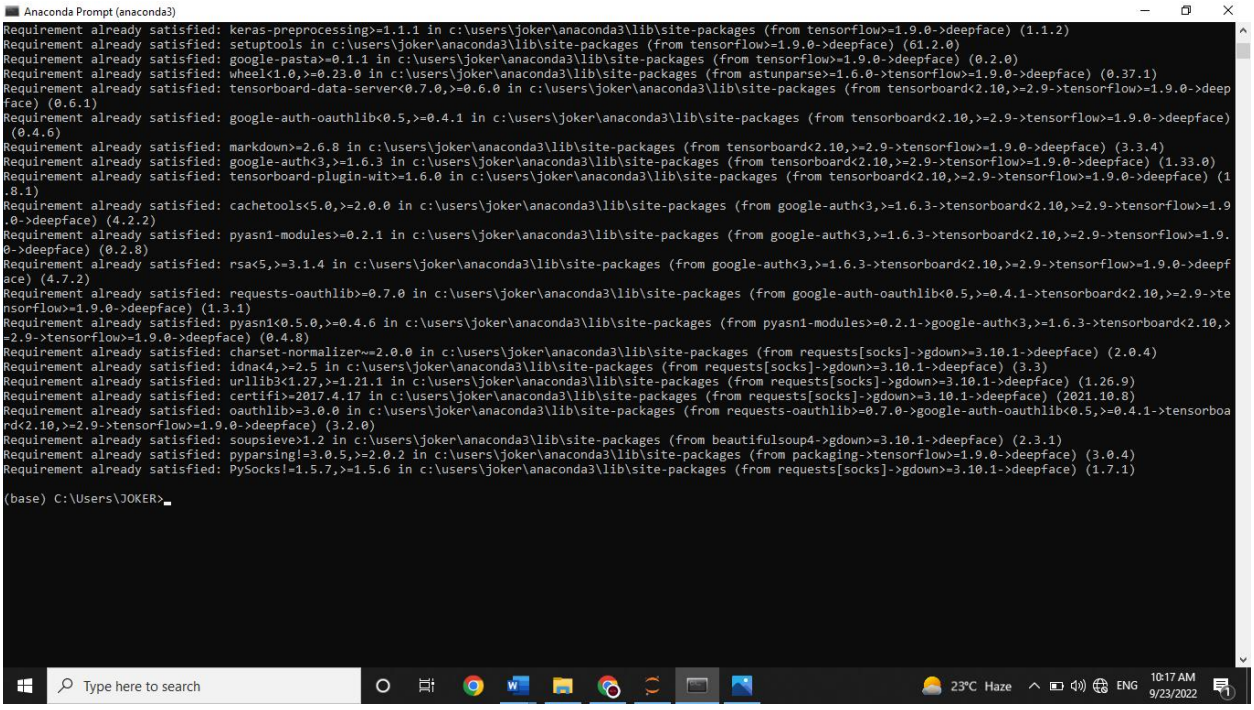
Note: If there is an any type of error copy the error message and search in the internet for solution because of this library is important for face recognition for both training and testing data.

- Where Opencv is used for the Processing the images and interact with user and system. It is meant to use for many types of image classification, pre-processing, Live interaction etc.,

4.4 INSTALLATION OF DEEPFACE LIBRARY:

1. Open the anaconda command prompt use the command “pip install deepface”





Note: This is important library for emotion detection it take little longer than other modules. Minimum 20 min.

2. Now open PC (local desktop\ file manager)→"Local Disk C:\" drive→"user"→open current desktop user folder (example: my desktop is named as "JOKER" so I am opening that folder)

In path form: "C:\Users\---desktop name---"

3. Create a new folder and name it as "**.deepface**" (dot{.} deepface)
4. And open that folder and create the new folder as "**weights**" and open it.
5. https://drive.google.com/file/d/1GnwEEk-Bh1YvCMAHcwKnerpUWqoD7s72/view?usp=sharing
6. From this link download the file it contains the trainned model of facial expression in format of .h5 file.
7. Move the file in the "weights" folder.

Congratulation you successfully installed deepface.

4.5 GETTING THE HAARCASCADE FILE:

1. Download this file make sure it should be in format of .xml only. https://drive.google.com/file/d/1ooAHbsixjwSuA6oFaAHrXtA-H0qQV0rH/view?usp=sharing

Congratulation for successful dumping of haarcascade file.

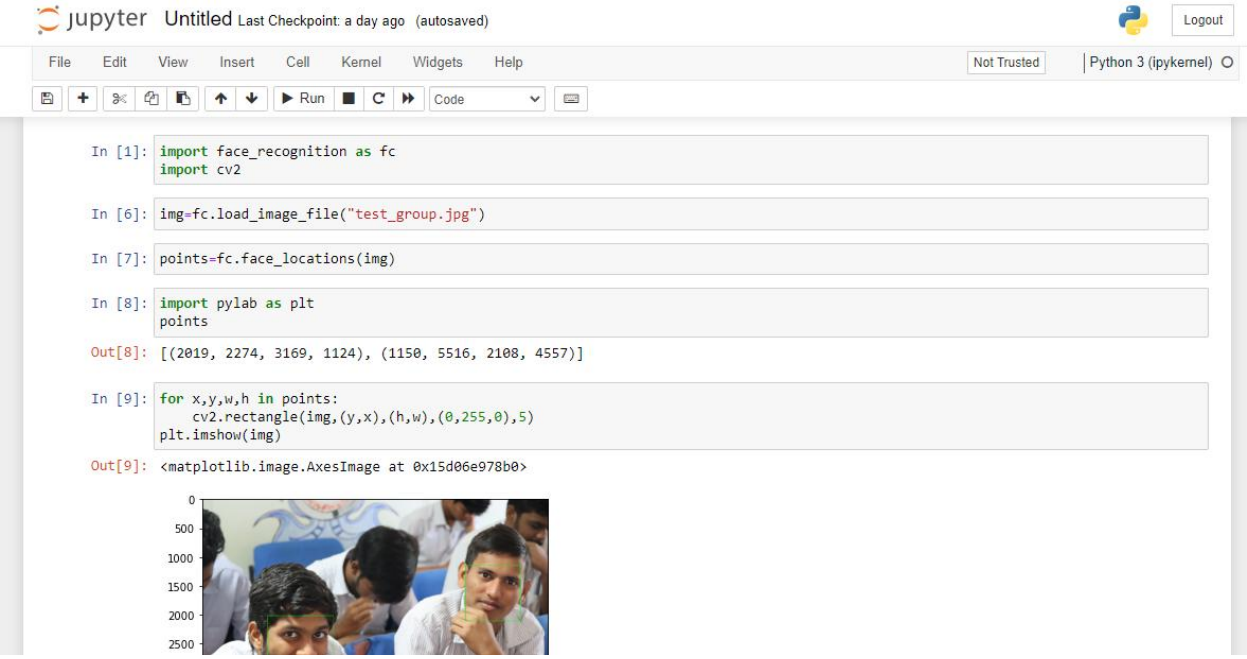
Note: this file is used for finding the faces which has accuracy of 97.55%

Up to this we had completed the all-system requirements.

5 DETECTING FACES USING FACE RECOGNITION LIBRARY

5.1 FROM PICTURES:

See the Pictures and follow:



In coding part following steps are involved:

1. Importing the face_recognition and opencv library to Jupyter notebook.
2. In code we use face_recognition as fc variable.
3. For loading an image, we use the predefined function as "fc.load_image_file("---name of file----")" to a variable "img".
4. For finding the location of images we use the pre-defined functions as" fc.loaction(--image--)" to a variable "point". Return the values which is according to the image which in order of from top to bottom.

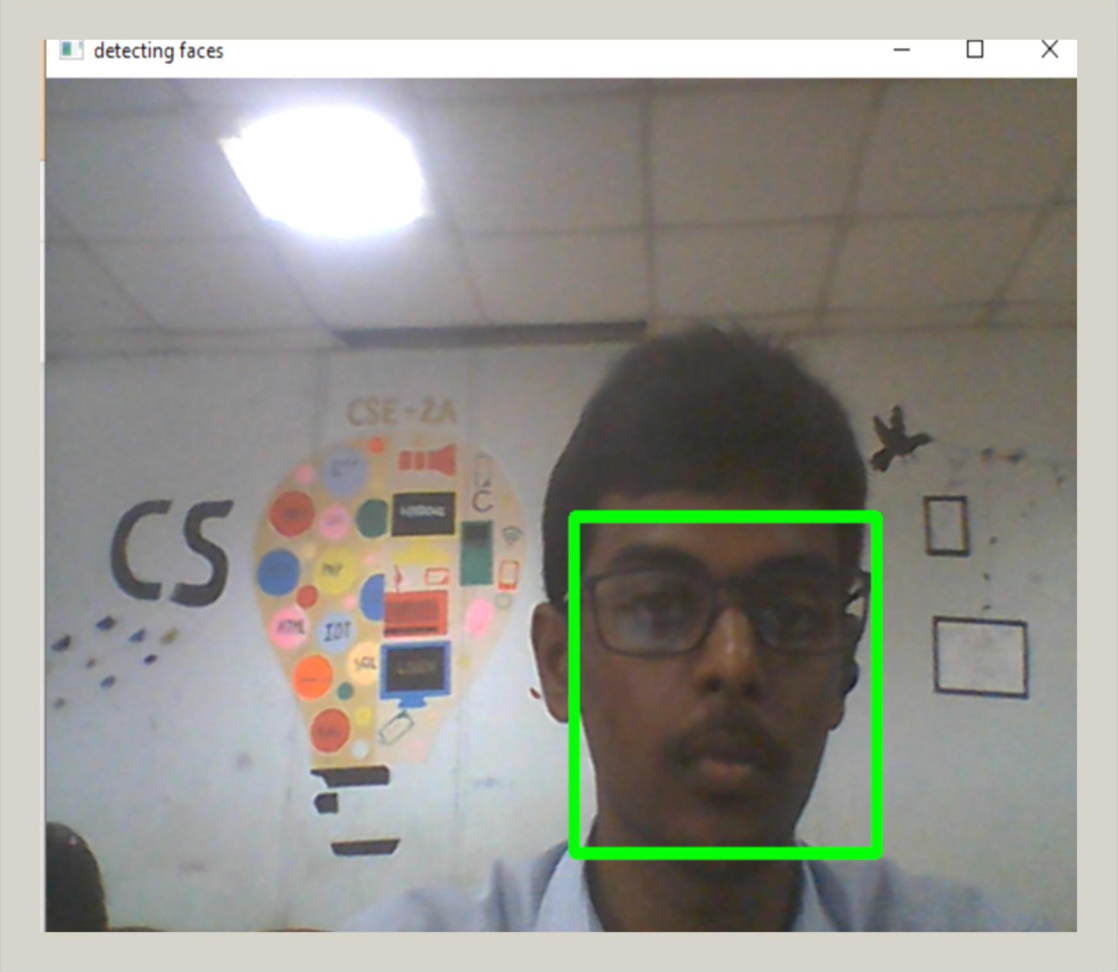
5. From the points it has the 4 coordinates which contain the face location as “x, y, w, h” in loop because if number of faces are more in image than we need to find the location each person so we use loop to find and draw the rectangle in the locations.

Note: In putting the rectangle it is referred as (x, y) as (y, x) which means it take y axis as first and then x axis as second for the image location.

6. Here we use the cv2.rectangle() function for put the rectangle box around the faces.

5.2 LIVE DEMO:

```
In [12]: import cv2
         import face_recognition

In [13]: cap=cv2.VideoCapture(0)  # activating the camara
         while True:         # for unlimited time to do process
             ret,frame=cap.read()   # returns the frame and frame number
             points=fc.face_locations(frame)  # used to locate the points
             for x,y,w,h in points:
                 cv2.rectangle(frame,(y,x),(h,w),(0,255,0),5)
             cv2.imshow('detecting faces',frame)   # return the frame after detection
             if cv2.waitKey(1) & 0xff==ord('q'):   # used for closing the loop by typing the "q" from keyboard
                 break
         cap.release()        # deacitvating the cam which we used now
         cv2.destroyAllWindows()   # closing the all related windows which are opened

In [ ]:
```



Here we are doing the live demo. For each line is explained respectively.

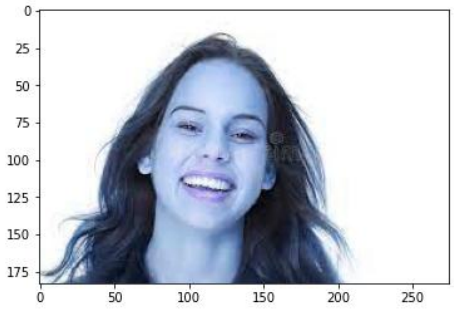
6 DETECTING FACES USING HAARCASCADE FILE

6.1 FROM THE IMAGES:

```
In [17]: import cv2
         import pylab as plt
         facecade=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
```

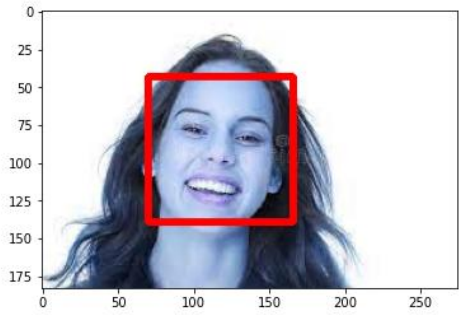
```
In [47]: img=cv2.imread("happy.jpg")  # here cv2 read image in the BGR format
         plt.imshow(img)
```

```
Out[47]: <matplotlib.image.AxesImage at 0x15d02576070>
```




```
In [41]: gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
         faces= facecade.detectMultiScale(gray,1.3,4)
         for (x,y,w,h) in faces:
             cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),4)
         plt.imshow(img)
```

```
Out[41]: <matplotlib.image.AxesImage at 0x15d02302130>
```



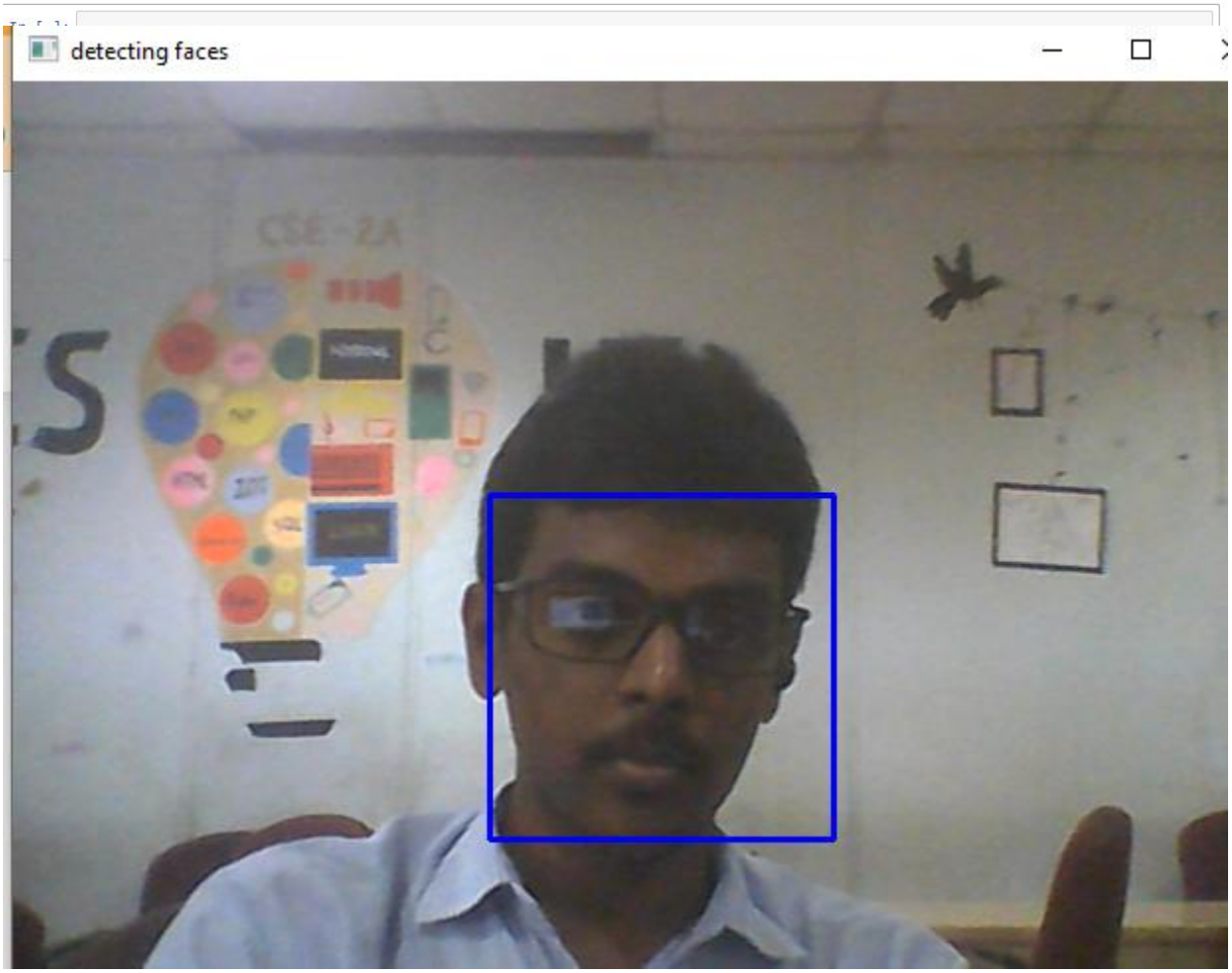
1. Here we used cascade file for detecting the faces and returns the 4 coordinate points similar to face recognition data.
2. For using the cascade file, it will detect the gray images fast than compared to the normal RGB images or BGR images.
3. In above code “gray” is used to convert the image into gray scale.
4. And “faces” is the variable which is used to store the coordinate points which is extracted from the image.
5. detectMultiScale give the number of faces present in the images.
6. Finally showing the detected images.

6.2 LIVE DEMO:

In [ ]:

In [52]:
```
import cv2
facecade=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
```

In [61]:
```
cap=cv2.VideoCapture(0)  # activating the camara
while True:         # for unlimited time to do process
    ret,frame=cap.read()   # returns the frame and frame number
    gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces= facecade.detectMultiScale(gray,1.2,4)
    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
    cv2.imshow('detecting faces',frame)   # return the frame after detection
    if cv2.waitKey(1) & 0xff==ord('q'):   # used for closing the loop by typing the "q" from keyboard
        break
cap.release()        # deacitvating the cam which we used now
cv2.destroyAllWindows()   # closing the all related windows which are opened
```

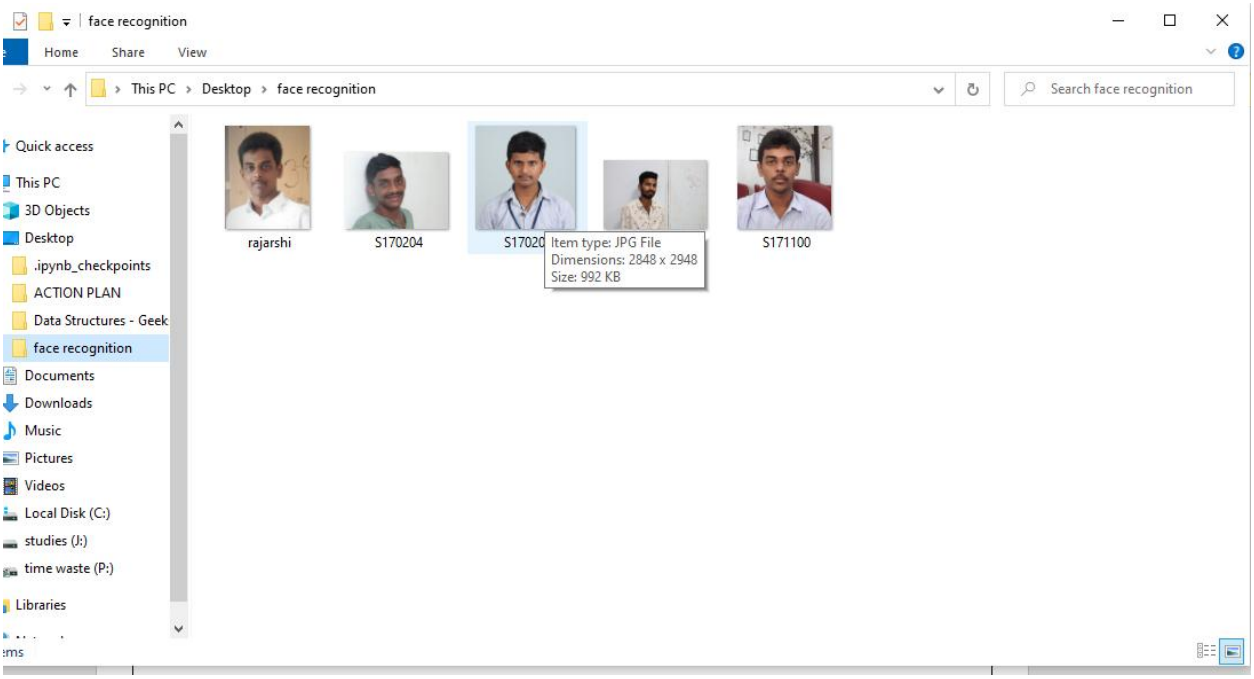


Here we do live demo for detection. It can make wrong detection due to scaleratio in the detection.

7 FACE RECOGNITION

7.1 ENCODED DATASETS

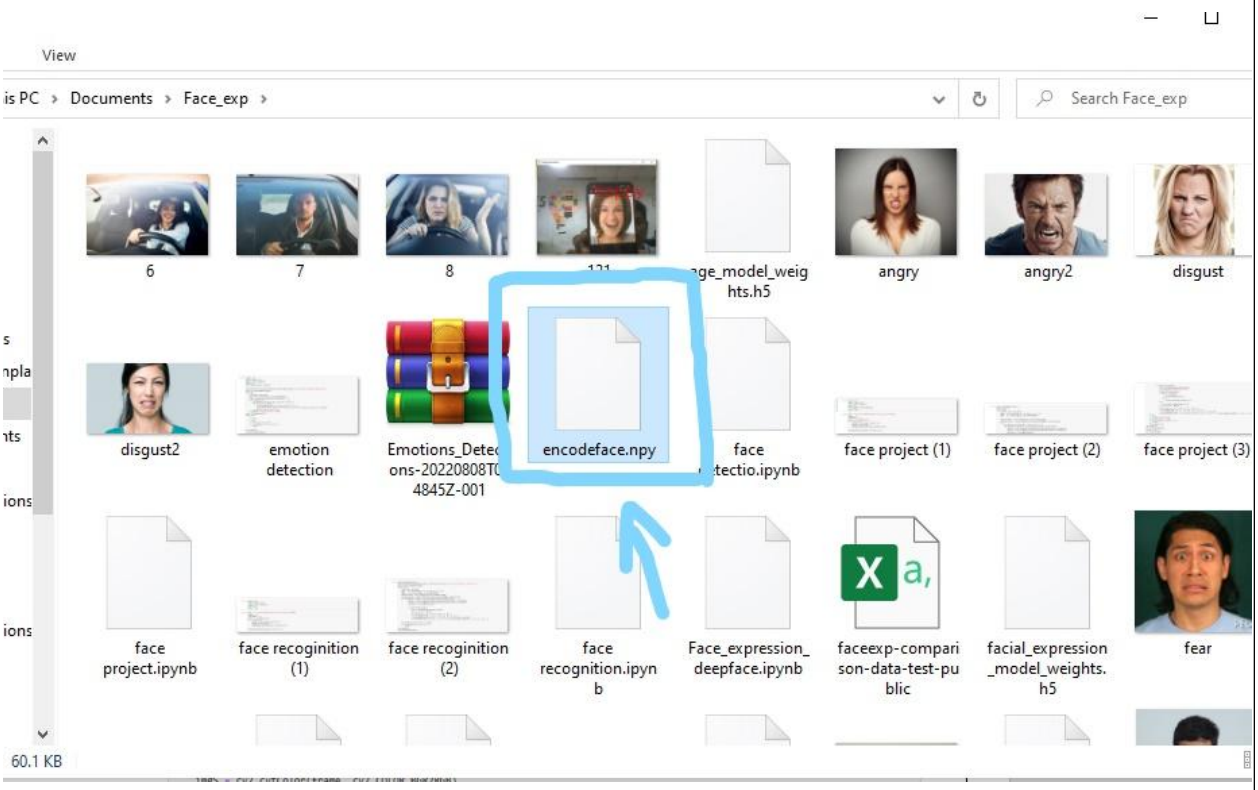
1. Firstly, we need to collect the images or datasets to recognize the person.
2. Here I am taking my images of random and stored in some location.



3. Now we need to encode the data and store in one location. Because no need to encoding again and again.

```
In [3]: def findEncodings(images):       #this block is used for the training the images in the digital format
            encodeList = []              # empty list for storing the encoded images for recognition
            for img in images:
                img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)           # converting images into original format
                encode = face_recognition.face_encodings(img)[0]       # storing the encoded images in encode
                encodeList.append(encode)                              # storing the all encoded images
            return encodeList
        encodeListKnown = findEncodings(images)                    # return the encoded data
        #np.save('encodeface.npy',encodeListKnown)   # here encoded data takes time so for reduce the time we save the encode data
```

4. And we are ready to detect the images.
5. We saved the code using "**np.save("encodeface.npy", encoded_data)**"
6. By this we reuse the encoded data again and again which gives less compilation time for execution.
7. To load that file we use the command "**variable_name=np.load("encodeface,npy")**"



7.2 LIVE DEMO:

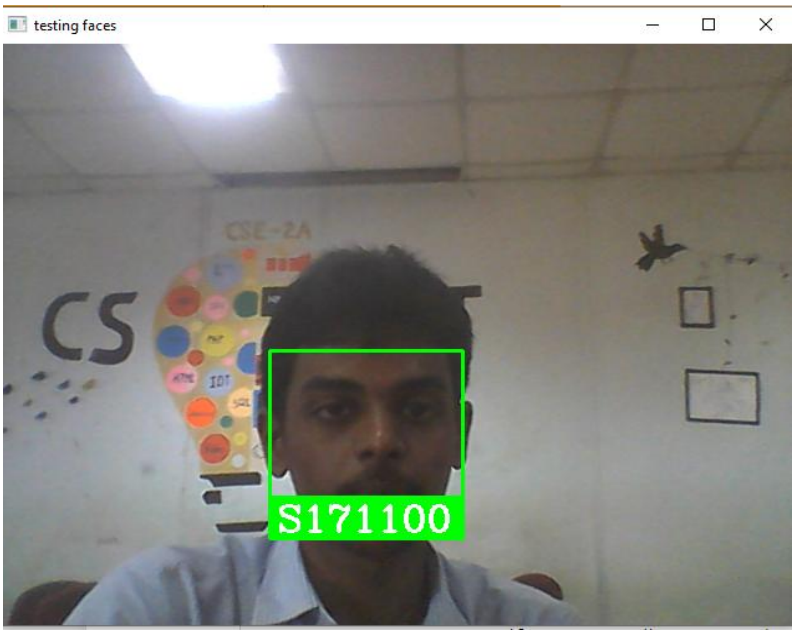
- You must need training dataset. Else it would not recognize.
- In below code we use encoded image which converted into digital format and saved in the given “encodListKnown” variable.
- And from this we can compare the faces using this encoded data and returns the result.
- In encode data contain face location as-well-as face landmarks means where the location of ears, eyes, mouth, nose, etc.,
- With this we can compare the faces and recognize faces also.

In [1]:
```
import cv2
import numpy as np
import face_recognition
import os
```

In [2]:
```
path = r'C:\Users\JOKER\Desktop\face recognition'  # here collected dataset that we used for recognition
images = []                                        # Only for images
classNames = []  # only for images names  note: image withrespective name of images
myList = os.listdir(path)  #taking the path access
for cl in myList:
    curImg = cv2.imread(f'{path}/{cl}')     #accessing the all images
    images.append(curImg)                     # storing the all images
    classNames.append(os.path.splitext(cl)[0])    # storing the names of the images

#encodeListKnown=np.load('encodeface.npy')    # loading the encoding file of the collected images it will store in array formate
```

In [3]:
```
def findEncodings(images):       #this block is used for the training the images in the digital format
    encodeList = []              # empty list for storing the encoded images for recognition
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)            # converting images into original format
        encode = face_recognition.face_encodings(img)[0]        # storing the encoded images in encode
        encodeList.append(encode)                               # storing the all encoded images
    return encodeList
encodeListKnown = findEncodings(images)                     # return the encoded data
#np.save('encodeface.npy',encodeListKnown)   # here encoded data takes time so for reduce the time we save the encode data
```

In [16]:
```
cap=cv2.VideoCapture(0)    # web cam or camara accessing command
while True:
    ret,frame= cap.read()                              # here frame is the photo for every secound
    imgS = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    facesCurFrame = face_recognition.face_locations(imgS)        # loacting the face in the frame
    encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)  # encoded the faces in the frame
    for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
            matches = face_recognition.compare_faces(encodeListKnown, encodeFace)   # compare the encoded face from cam and encod
            faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)  # find the which position of the faces is matc
            matchIndex = np.argmin(faceDis)  # convert the array to normal number such as integer format

            if matches[matchIndex]:
                name = classNames[matchIndex].upper()   # finding the name via using the matchIndex number
                x,y,w,h = faceLoc        # loacation of the faces which are matched in x/4 ratio format
                cv2.rectangle(frame, (y,x),(h,w), (0, 255, 0), 2)    # creating the rectangle on matched faces
                cv2.rectangle(frame, (y,w-35),(h,w), (0, 255, 0), cv2.FILLED)  # this is used for putting the recognized name
                # box for face and box for text
                cv2.putText(frame, name, (h + 6, w - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)  # name text

    cv2.imshow('testing faces',frame)   # return the frame after recognition
    if cv2.waitKey(1) & 0xff==ord('q'):   # used for closing the loop
        break
cap.release()        # deacitvating the cam which we used now
cv2.destroyAllWindows()   # closing the all related windows which are opened
```



8 EMOTION DETECTION

- As for this part we need to train the different types of emotions. And we can use that training data to predict the emotion of the images.
- Although for training the different types of images takes more than expected time so we use the already trained data which we seen in the deepface library installation.
- The trained images are known to be weights of that images.

8.1 FROM PICTURE TO DETECT THE EMOTIONS:

Steps involved in this process:

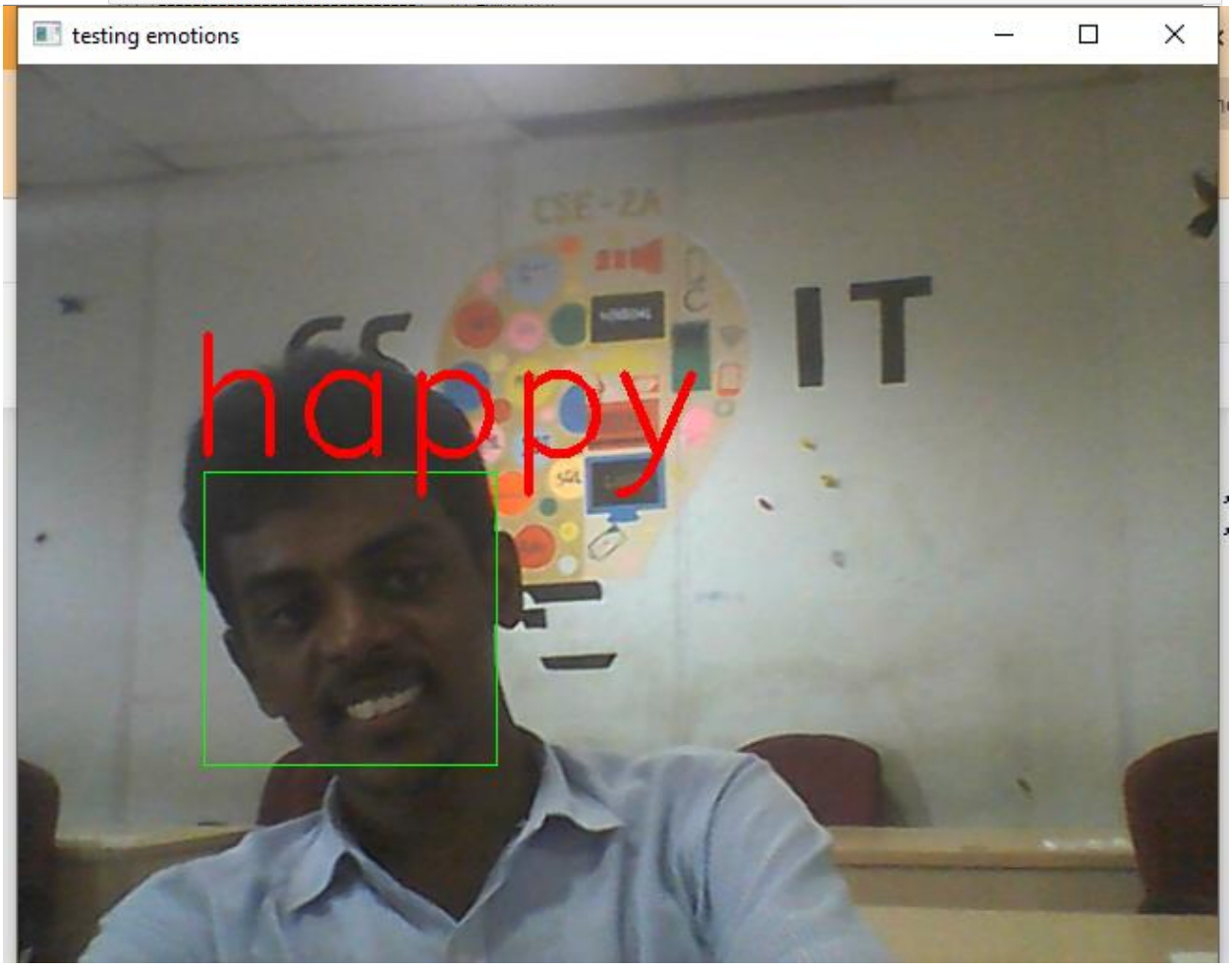
1. Load the image using opencv python. And store in a variable.
2. Load deepface library as "**from deepface import DeepFace as dp**" in notebook.
3. By using command "**dp.analyze(image, actions=['emotion'])**"
4. It returns the all-emotions ratio such as happy, sad, angry, surprise, fear, disgust, neutral.
5. From all of this we only need the dominate emotion.
6. The results are stored in the dictionary datatype.

```
In [16]: import cv2
         from deepface import DeepFace as dp
         import pylab as plt

In [17]: img= cv2.imread("sad.jpg")
         result=dp.analyze(img,actions=['emotion'])
         print(result)

         1/1 [==============================] - 0s 63ms/step
         {'emotion': {'angry': 1.9884256646037102, 'disgust': 1.746329858498541e-07, 'fear': 0.06470748921856284, 'happy': 0.000425765665
         55681785, 'sad': 94.09490823745728, 'surprise': 3.0651909899148677e-07, 'neutral': 3.8515325635671616}, 'dominant_emotion': 'sa
         d', 'region': {'x': 509, 'y': 243, 'w': 484, 'h': 484}}

In [18]: faceCascade=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
         gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
         faces= faceCascade.detectMultiScale(gray,1.1,4)
         for (x,y,w,h) in faces:
             cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),7)
             cv2.putText(img, result['dominant_emotion'],(x-10,y-10),cv2.FONT_HERSHEY_SIMPLEX,10,(0,0,255),cv2.LINE_4)
         plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))
```


sad
BIGSTOCK
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400

8.2 LIVE DEMO:

In [26]:
```
import cv2
from deepface import DeepFace as dp
faceCascade=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
font=cv2.FONT_HERSHEY_SIMPLEX
def exp():
    ret,frame= cap.read()
    gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces= faceCascade.detectMultiScale(gray,1.1,4)
    if(len(faces)!=0):
        res=dp.analyze(frame, actions=['emotion'])
        for (x,y,w,h) in faces:
            cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),1)
            cv2.putText(frame, res['dominant_emotion'],(x-10,y-10),font,3,(0,0,255),cv2.LINE_4)
        print(res['dominant_emotion'])
    cv2.imshow('testing emotions',frame)


cap=cv2.VideoCapture(0)
while True:
    try:
        exp()
    except:
        print("no face found")
    if cv2.waitKey(1) & 0xFF==ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```



9 FACE RECOGNITION AND EMOTION DETECTION

Here we combined both the codes which means when face is recognized then it will detect the emotions.

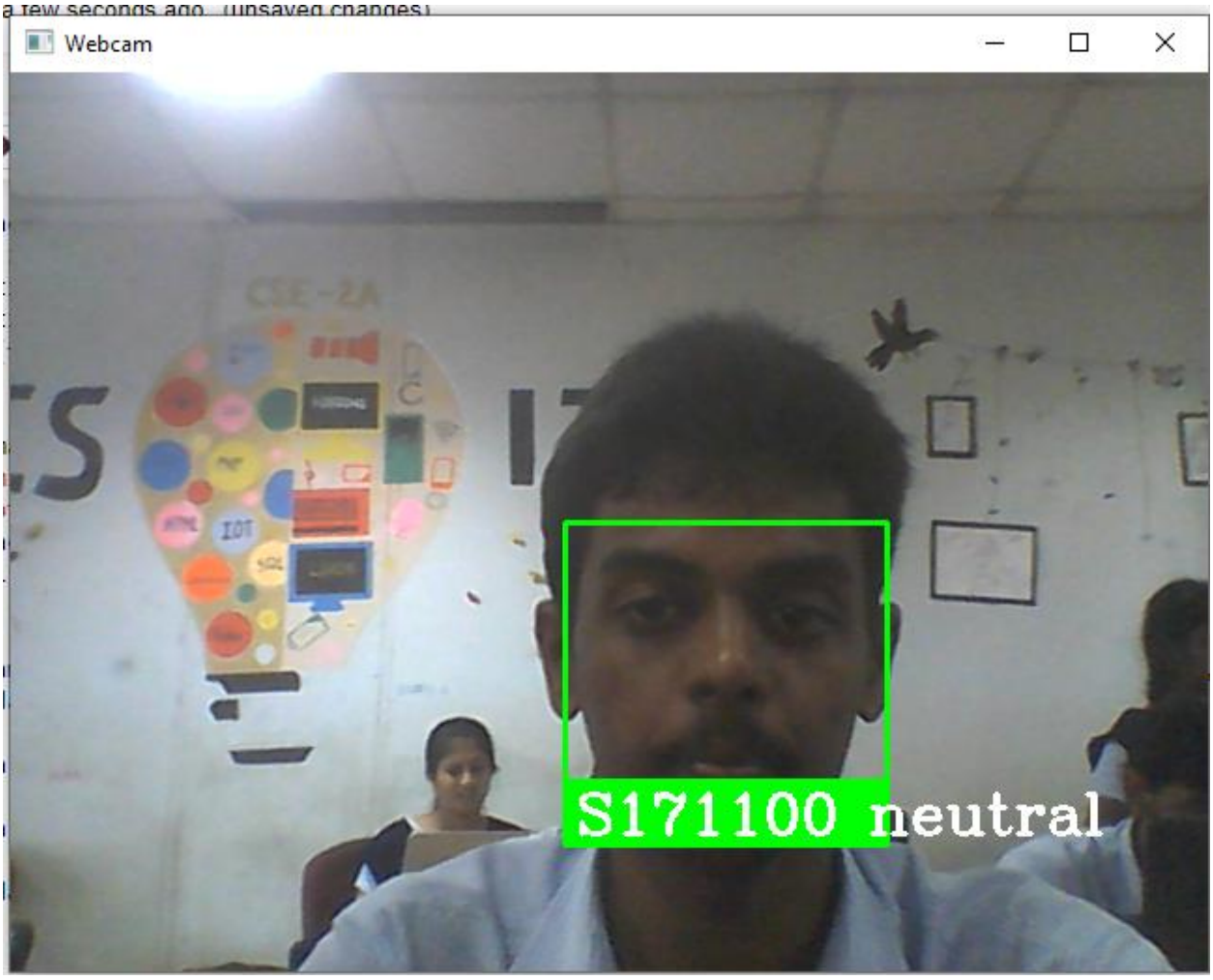
In [3]:
```
cap = cv2.VideoCapture(0)
record_data={}
while True:
    success, img = cap.read()
    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)
    try:
        res=DeepFace.analyze(img, actions=['emotion'])
        for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
            matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
            faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
            matchIndex = np.argmin(faceDis)
```

In [1]:
```
import cv2
import numpy as np
import face_recognition
import os
from deepface import DeepFace
import time
```

In [2]:
```
path = r'C:\Users\JOKER\Documents\Face_exp\CSE-1A PHOTOS'
images = []
classNames = []
myList = os.listdir(path)
for cl in myList:
    curImg = cv2.imread(f'{path}/{cl}')
    images.append(curImg)
    classNames.append(os.path.splitext(cl)[0])

encodeListKnown=np.load('encodeface.npy')
```

```
            if matches[matchIndex]:
                name = classNames[matchIndex].upper()
                print(name,res['dominant_emotion'])
                x=res['dominant_emotion']
                if name in record_data:
                    if x in a[name]:
                        record_data[name][x]+=1
                    else:
                        record_data[name].update({x:1})
                else:
                    record_data.update({name:{x:1}})
                y1, x2, y2, x1 = faceLoc
                y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
                cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
                cv2.putText(img, name+' '+res['dominant_emotion'], (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255)
    except:
        print('no face found')
    cv2.imshow('Webcam', img)
    if cv2.waitKey(1) & 0xff==ord('q'):
        break
    time.sleep(1.200)
cap.release()
cv2.destroyAllWindows()
print('success')
print(record_data)
```

Output:


Webcam
S171100 neutral