Q1.Use python and opencv to implement a simple face detection algorithm, it should detect face on a livecamera feed (webcam) and save any faces captured in a folder. (Bonus: beep when a face is detector)

Code:

# Creating database

# It captures images and stores them in datasets

# folder under the folder name of sub\_data

import cv2, sys, numpy, os

haar\_file = 'haarcascade\_frontalface\_default.xml'

# All the faces data will be

# present this folder

datasets = 'datasets'

# These are sub data sets of folder,

# for my faces I've used my name you can

# change the label here

sub\_data = 'Mohit Pachauri'

path = os.path.join(datasets, sub\_data)

if not os.path.isdir(path):

os.mkdir(path)

# defining the size of images

(width, height) = (130, 100)

#'0' is used for my webcam,

# if you've any other camera

# attached use '1' like this

face\_cascade = cv2.CascadeClassifier(haar\_file)

webcam = cv2.VideoCapture(0)

# The program loops until it has 30 images of the face.

count = 1

while count < 30:

(\_, im) = webcam.read()

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

faces = face\_cascade.detectMultiScale(gray, 1.3, 4)

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

cv2.rectangle(im, (x, y), (x + w, y + h), (255, 0, 0), 2)

face = gray[y:y + h, x:x + w]

face\_resize = cv2.resize(face, (width, height))

cv2.imwrite('% s/% s.png' % (path, count), face\_resize)

count += 1

cv2.imshow('OpenCV', im)

key = cv2.waitKey(10)

if key == 27:

break

# It helps in identifying the faces

import cv2, sys, numpy, os

size = 4

haar\_file = 'haarcascade\_frontalface\_default.xml'

datasets = 'datasets'

# Part 1: Create fisherRecognizer

print('Recognizing Face Please Be in sufficient Lights...')

# Create a list of images and a list of corresponding names

(images, lables, names, id) = ([], [], {}, 0)

for (subdirs, dirs, files) in os.walk(datasets):

for subdir in dirs:

names[id] = subdir

subjectpath = os.path.join(datasets, subdir)

for filename in os.listdir(subjectpath):

path = subjectpath + '/' + filename

lable = id

images.append(cv2.imread(path, 0))

lables.append(int(lable))

id += 1

(width, height) = (130, 100)

# Create a Numpy array from the two lists above

(images, lables) = [numpy.array(lis) for lis in [images, lables]]

# OpenCV trains a model from the images

# NOTE FOR OpenCV2: remove '.face'

model = cv2.face.LBPHFaceRecognizer\_create()

model.train(images, lables)

# Part 2: Use fisherRecognizer on camera stream

face\_cascade = cv2.CascadeClassifier(haar\_file)

webcam = cv2.VideoCapture(0)

while True:

(\_, im) = webcam.read()

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

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

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

cv2.rectangle(im, (x, y), (x + w, y + h), (255, 0, 0), 2)

face = gray[y:y + h, x:x + w]

face\_resize = cv2.resize(face, (width, height))

# Try to recognize the face

prediction = model.predict(face\_resize)

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

if prediction[1]<500:

cv2.putText(im, '% s - %.0f' %

(names[prediction[0]], prediction[1]), (x-10, y-10),

cv2.FONT\_HERSHEY\_PLAIN, 1, (0, 255, 0))

else:

cv2.putText(im, 'not recognized',

(x-10, y-10), cv2.FONT\_HERSHEY\_PLAIN, 1, (0, 255, 0))

cv2.imshow('OpenCV', im)

key = cv2.waitKey(10)

if key == 27:

break