**Training Phase :**

*import face\_recognition*

*import os*

*import pickle*

*from sklearn import neighbors*

*encodings = []*

*names = []*

**#train directory**

*train\_dir = ("C:\\Users\\91805\\Desktop\\PSG\\PSG - SEMESTER 3\\PROJECT WORK SEM III\\project\\train\_data")*

*train\_dir\_path = "C:\\Users\\91805\\Desktop\\PSG\\PSG - SEMESTER 3\\PROJECT WORK SEM III\\project\\train\_data"*

**#enumerate throughout training directory**

*count = 0*

*for subdir, dirs, files in os.walk(train\_dir\_path):*

*for file in files:*

*image\_path = os.path.join(subdir, file)*

*if (image\_path.find(".jpeg") == -1):*

*continue*

*path\_cmp = image\_path.split("\\")*

*person = path\_cmp[len(path\_cmp) - 2]*

#print(len(path\_cmp))

#print(image\_path)

#print(person)

**# Loop through each training image for the current person**

**# Get the face encodings for the face in each image file**

*face = face\_recognition.load\_image\_file(image\_path)*

*face\_bounding\_boxes = face\_recognition.face\_locations(face)*

*print("processing " + str(count) + " " + image\_path)*

**#If training image contains exactly one face**

*if len(face\_bounding\_boxes) == 1:*

*face\_enc = face\_recognition.face\_encodings(face)[0]*

**# Add face encoding for current image with corresponding label (name) to the training data**

*encodings.append(face\_enc)*

*names.append(person)*

*print("name : " + person)*

*else:*

*print(person + "/" + person + " was skipped and can't be used for training")*

*count = count + 1*

**# Create and train the KNN classifier**

*knn\_clf = neighbors.KNeighborsClassifier(weights='distance')*

*knn\_clf.fit(encodings, names)*

**# save the model to disk**

*print("saving model to disk")*

*modelfilename = 'knn\_finalized\_model.sav'*

*pickle.dump(knn\_clf, open(modelfilename, 'wb'))*

*print("saved model to disk in file " + modelfilename)*

**# save the model**

**Testing Phase :**

import face\_recognition

from sklearn import neighbors

import os

test\_dir\_path = "C:\\Users\\91805\\Desktop\\PSG\\PSG - SEMESTER 3\\PROJECT WORK SEM III\\project\\test\_data"

output\_dir = "C:\\Users\\91805\\Desktop\\PSG\\PSG - SEMESTER 3\\PROJECT WORK SEM III\\project\\out\_dir"

import pickle

from PIL import Image, ImageDraw

import numpy as np

def draw\_name\_on\_face(image\_path, person):

known\_image = face\_recognition.load\_image\_file(image\_path)

# Find all the faces and face encodings in the unknown image

face\_locations = face\_recognition.face\_locations(known\_image)

face\_encodings = face\_recognition.face\_encodings(known\_image, face\_locations)

# Convert the image to a PIL-format image so that we can draw on top of it with the Pillow library

# See http://pillow.readthedocs.io/ for more about PIL/Pillow

pil\_image = Image.fromarray(known\_image)

# Create a Pillow ImageDraw Draw instance to draw with

draw = ImageDraw.Draw(pil\_image)

# Loop through each face found in the unknown image

for (top, right, bottom, left), face\_encoding in zip(face\_locations, face\_encodings):

# See if the face is a match for the known face(s)

#matches = face\_recognition.compare\_faces(known\_face\_encodings, face\_encoding)

name = person[0].replace("pins\_","")

print(name)

# If a match was found in known\_face\_encodings, just use the first one.

# if True in matches:

# first\_match\_index = matches.index(True)

# name = known\_face\_names[first\_match\_index]

# Or instead, use the known face with the smallest distance to the new face

#face\_distances = face\_recognition.face\_distance(known\_face\_encodings, face\_encoding)

#best\_match\_index = np.argmin(face\_distances)

#if matches[best\_match\_index]:

# name = known\_face\_names[best\_match\_index]

left = left -10

right = right + 10

bottom = bottom + 25

top = top - 20

# Draw a box around the face using the Pillow module

draw.rectangle(((left, top), (right, bottom)), outline=(0, 0, 255))

# Draw a label with a name below the face

text\_width, text\_height = draw.textsize(name)

draw.rectangle(((left, bottom - text\_height - 10), (right, bottom)), fill=(0, 0, 255), outline=(0, 0, 255))

draw.text((left + 6, bottom - text\_height - 5), name, fill=(255, 255, 255, 255))

# Remove the drawing library from memory as per the Pillow docs

del draw

# Display the resulting image

pil\_image.show()

# You can also save a copy of the new image to disk if you want by uncommenting this line

#get image name

image\_file\_array = image\_path.split("\\")

image\_file\_name = image\_file\_array[len(image\_file\_array) - 1]

pil\_image.save(output\_dir + image\_file\_name)

modelfilename = 'knn\_finalized\_model.sav' #for svm its finalized\_model.sav

clf = pickle.load(open(modelfilename, 'rb'))

# Load the test image with unknown faces into a numpy array

for subdir, dirs, files in os.walk(test\_dir\_path):

for file in files:

image\_path = os.path.join(subdir, file)

if (image\_path.find(".jpeg") == -1):

continue

test\_image = face\_recognition.load\_image\_file(image\_path)

# Find all the faces in the test image using the default HOG-based model

face\_locations = face\_recognition.face\_locations(test\_image)

no = len(face\_locations)

print("Number of faces detected: ", no)

# Predict all the faces in the test image using the trained classifier

print("Found faces in image " + image\_path)

for i in range(no):

test\_image\_enc = face\_recognition.face\_encodings(test\_image)[i]

name = clf.predict([test\_image\_enc])

draw\_name\_on\_face(image\_path, name)

#print(\*name)















