A Project Report on

SMART ATTENDANCE SYSTEM

Submitted by

B. HARIKA – R180503

S. KRISHNAVENI – R180893

Under the guidance of

M. MUNI BABU

M.Tech, (Ph.D.), Assistant Professor

Department of Computer Science and Engineering

Rajiv Gandhi University of Knowledge Technologies
RK Valley, Kadapa (Dist), Andhra Pradesh, 516330

Rajiv Gandhi University of Knowledge Technologies
RK Valley, Kadapa (Dist), Andhra Pradesh, 516330

CERTIFICATE

This is to certify that the project work titled **"SMART ATTENDANCE SYSTEM"** is abonafied project work submitted by **B.HARIKA – R180503**, **S.KRISHNAVENI - R180893** in the department of COMPUTER SCIENCE AND ENGINEERING in partial fulfilment of requirements for the award of degree of Bachelor of Technology in **Computer Science and Engineering** for the year 2023- 2024 carried out the work under the supervision.

GUIDE	HEAD OF THE DEPARTMENT
M. MUNI BABU	N. SATYANANDARAM
M. Tech, (Ph. D)	PGDIT, MSIT
Assistant Professor.	IIIT, Hyderabad.

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.

I am extremely grateful to our respected Director, Prof. K. SANDHYA RANI for fostering an excellent academic climate in our institution.

I also express my sincere gratitude to our respected Head of the Department Mr. N. SATHYANANDARAM for his encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project.

I would like to convey thanks to our guide at college Mr. M. MUNI BABU for his guidance, encouragement, co-operation and kindness during the entire duration of the course and academics.

My sincere thanks to all the members who helped me directly and indirectly in the completion of project work. I express my profound gratitude to all our friends and family members for their encouragement.

Table of Contents

Abstract:

Face Recognition is a computer application that is capable of identifying and verifying human faces from an image using a digital camera. Face Recognition can be practically implemented in real-life environment as an automatic attendance management system. Varying lighting condition, noise in face images, scale and pose are the issues and variations in human facial appearance. Face Recognition is to look at the main features of a captured face and compare them to features on other faces that have been stored in the database. The programming language used to build the smart attendance system is machine learning with python. The smart attendance system has gained significant attention in recent years due to its potential applications in various domains. This project focuses on implementing face recognition for attendance management. Smart attendance system will use facial features such as to identify individuals and automate the attendance recording process.

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

Face recognition for attendance is a system that uses cameras and computer programs to identify and record when people are present. Instead of using traditional methods like signing a paper or using a card, the system recognizes each person's unique facial features to mark their attendance. It's like having a virtual assistant that knows who you are just by looking at your face and keeps track of when you arrive or leave a place. This makes the process faster and more accurate, reducing the need for manual record-keeping.

Face recognition provides a seamless and convenient user experience. User do not need to remember passwords or carry physical tokens, they can simply use their faces for authentication. It can process and match faces in real-time, making it suitable for scenarios that require quick identification, such as law enforcement, crowd management and customer service.

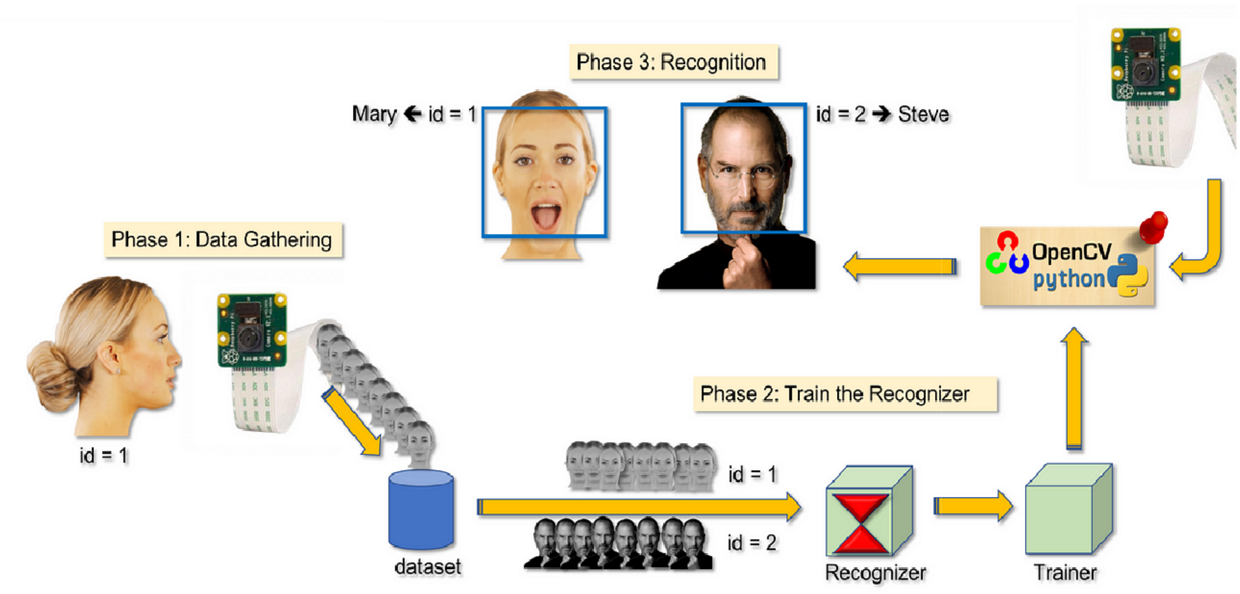


Figure 1.1: Complete process

As shown in the figure 1.1, the given data gathering, create a dataset, train the recognizer, train it and recognition is done in the complete process of smart attendance system.

Problem Statement:

In existing systems of biometric like fingerprints, passwords, when we compare with those face recognition is faster, secure and convenient. In [1, 2] lock/unlock are very inefficient. There may be possible of losing keys or codes or passwords. So, we propose a face recognition system that can be able to recognize faces with maximum accuracy possible. Traditional attendance management methods, such as manual sign-in sheets or swipe cards, are time-consuming, error-prone, and susceptible to fraud. There is a need for a more efficient and reliable system to automate attendance recording. Face recognition system provides a potential solution to this problem by accurately identifying individuals based on their facial features.

II. LITERATURE REVIEW

Table 2.1: Literature Review

Year	Author	Method	Outcome	Drawback
2023	Garcia et al.	***Hybrid Approach:*** It is a combination of two or more computational techniques which provide more advantages to detect components than any other individual technique. It help to improve data analysis. The hybrid technique helps qualitative research to be effective.	Achieved real-time performance	Limited accuracy for low-resolution images
2022	Wang et al.	***Capsule Networks:*** In the Capsule networks, specific methodology is applied to image processing to try to affect an understanding of objects from a three -dimensional spectrum.	Robust to occlusion	Longer traning time
2021	Lee et al.	***Attention Mechanism:*** It is an input processing technique of netural networks. This machanism helps neural networks	Enhanced feature representation	Sensitivity to pose variations

		solve complicated tasks by dividing them into smaller areas of attention and processing them sequentially.		
2020	Patel et al.	***Convolutional GAN:*** This GAN uses a deep convolutional neural network for producing high-resolution image generation that can be differentaiated. Convolutios are a technique for drawing out important information from the generated data.	Generated realistic faces	Difficulty in fine-tuning
2019	Johnson et al.	***Siamese Network:*** It consists of twin networks which accept distinct inputs but are joined by an energy function at the top. This function computes a metric between the highest level feature representation on each side.	Improved generalization	Relatively small dataset
2018	Smith et al.	***Deep Face Net:*** Face Net takes an image of a face as input and outputs the embedding vector.	Achieved 98% accuracy	High computational cost

Motivation:

The motivation behind face recognition is to streamline attendance management processes in various settings, including educational institutions, workplaces, and events. By utilizing face recognition system, the system aims to improve efficiency, accuracy, and security in attendance tracking. The motivation also lies in reducing manual effort, eliminating the need for physical identification cards, and providing a seamless user experience. Less computational cost, relatively large dataset, short training time, un-limited accuracy for low-resolution images. Face detection helps with facial analysis as well. It helps to figure out which parts of a video or picture should be focused on to determine gender, age, or feelings. In the same way, face detection data is built into the algorithms of facial recognition systems, which create "face print" maps of facial features. Face detection assists in identifying the elements of the video or image that are necessary to generate a face print.

Contribution:

The contribution of face recognition is Improves security and improves surveillance efforts and helps track down criminals and terrorists. Personal security is enhanced when users use their faces in place of passwords, because there is nothing for hackers to steal or change. In the past, identification was manually performed by a person, this was inefficient and frequently inaccurate. Face detection allows the identification process to be automated, saving time and increasing accuracy. It involves the integration of computer vision techniques, machine learning algorithms, and efficient database management. The system's contribution lies in automating attendance recording, reducing administrative overhead, and improving overall efficiency and accuracy in attendance management.

Haar cascade is an algorithm that can detect objects in images, irrespective of their scale in image. This algorithm is not so complex and can run in real-time. We can train a haar-cascade detector to detect various faces.

Algorithm:

1. ***Haar – like Features Extraction***:
    - The algorithm uses Haar like features, which are simple rectangular filters, to represent different image features, such as edges, lines, and textures. These features help distinguish between different objects in the image.
    - Haar like features are applied to the integral image of the original input image. The integral image enables fast computation of rectangular sum operations.

2. ***Training Data Collection:***
    - A large dataset of positive samples (images containing faces) and negative samples (images without faces) is required for training the algorithm. These samples should be label accordingly.
    - For face detection, positive samples will include images of faces, while negative samples will contain various background images.
3. ***Creating a Haar Cascade Classifier:***
    - The algorithm creates a cascade classifier consisting of multiple stages, each containing multiple weak classifiers (classifiers based on Haar-like features).
    - The classifiers are organized in a cascade, where the first stages are trained to be very fast in rejecting non-face regions, and later stages are progressively more complex to handle difficult cases.
4. ***Training the Cascade Classifier:***
    - The cascade classifier is trained using the AdaBoost (Adaptive Boosting) algorithm. AdaBoost selects the best Haar-like features and assigns.
    - weights to weak classifiers based on their performance in classifying positive and negative samples.
    - During training, the algorithm adjusts the weights of misclassified samples to emphasize hard-to-detect regions and improving overall accuracy.
5. ***Cascade of Classifiers:***
    - The cascade structure allows the algorithm to quickly reject non-face regions in the image and reducing the computational burden.

- At each stage of the cascade, a decision is made whether the region is likely to contain a face or not. If the region passes a stage, it proceeds to the next stage, and if it fails at any stage, it is rejected.

6. ***Sliding Window Detection:***
   - The trained cascade classifier is applied to the image using a sliding window approach. A window of fixed size moves across the image in a grid-like fashion, examining different regions of the image for potential faces.
   - At each window position, the Haar-like features are computed within the window, and the cascade classifier evaluates whether the region contains a face or not.
7. ***Non-maximum Suppression:***
   - After the detection step, multiple overlapping detections may be found. To eliminate duplicate detections, non-maximum suppression is used.
   - This process keeps only the most confident detection in a given region and suppresses all other overlapping detections with lower confidence scores.
8. ***Face Recognition:***
   - Once the faces are detected in the image, additional steps may be performed for face recognition, such as feature extraction, face alignment and comparison with a database of known faces.

Advantages:

1. ***Fast Detection:*** Haar cascades are computationally efficient and can achieve real-time or near real-time face detection on standard hardware. The cascading nature of the algorithm allows it to quickly reject non-face regions in an image, reducing the search space and speeding up the process.
2. ***Accuracy:*** While Haar cascades might not be the most accurate face recognition method available today, they can still achieve reasonably good results in various scenarios. The accuracy can be further improved by using multiple cascades or combining Haar cascades with other techniques.
3. ***Memory Efficient:*** Once the Haar cascade classifiers are trained, the model's memory footprint is relatively small compared to more complex deep learning models used in face

recognition. This makes it more feasible to deploy Haar cascades on resource-constrained devices.

4. ***Simple Implementation:*** Haar cascades are relatively easy to implement and use. Libraries like Open CV provide built-in support for Haar cascades and making them accessible to developers without extensive machine learning expertise.

5. ***Training Data:*** While the training process for Haar cascades does require label data, it doesn't need as much training data as deep learning approaches. This can be advantageous when dealing with limited datasets.

6. ***Real-time Application:*** The speed and efficiency of Haar cascades make them well-suited for real-time face recognition applications, such as video surveillance, facial expression analysis and human-computer interaction systems.

Disadvantages:

1. ***Accuracy and Performance:*** While Haar cascades can work well for simple face recognition tasks, they may struggle to achieve high accuracy and robustness in complex scenarios. More advanced methods like deep learning based approaches (e.g., CNNs) have generally surpassed Haar cascades in terms of accuracy.

2. ***Limited Pose and Scale Invariance:*** Haar cascades are not very effective at handling faces with extreme poses, rotations or varying scales. This limitation can lead to difficulties in recognizing faces that are not directly facing the camera or appear at different sizes.

3. ***Background Interference:*** Haar cascades rely on specific features (e.g., edges, corners) to identify objects, and they may confuse faces with similar patterns in the background. This can lead to false positives and incorrect face detections.

4. ***Training Complexity:*** Training Haar cascades requires collecting positive and negative samples, which can be a time-consuming process. Tuning the cascade parameters for optimal performance also demands expertise.

5. ***Resource Intensive:*** While Haar cascades were considered efficient at the time of their introduction, modern deep learning-based methods can offer more accuracy with less computational overhead.

6. ***Difficulty with Occlusions:*** Haar cascades struggle to recognize faces partially occluded by objects or hands, as the integral image-based approach they use may not handle occlusions well.

7. ***Gender and Race Bias:*** The accuracy of Haar cascades might be influenced by the gender and racial bias present in the training data, potentially leading to uneven performance across different demographics.
8. ***Maintenance and Updates:*** Haar cascades require periodic maintenance and updates to adapt to new face variations and overcome limitations. Deep learning-based methods, on the other hand, can learn continuously from new data without requiring the same level of manual intervention.

Applications:
1. ***Face Detection:*** One of the most popular applications of Haar cascades is face detection. They can efficiently detect faces in images or video streams, making them essential in applications such as facial recognition, emotion analysis and face tracking.
2. ***Object Detection:*** Apart from faces, Haar cascades can be trained to detect other objects like cars, pedestrians or specific items in images and videos. This makes them valuable in surveillance, autonomous vehicles and robotics.
3. ***Gesture Recognition:*** Haar cascades can be employed in recognizing hand gestures and movements. This application is commonly used in human-computer interaction systems and sign language recognition.
4. ***Facial Expression Recognition:*** By training Haar cascades on datasets with label facial expressions and it's possible to build systems that recognize emotions expressed on people's faces.
5. ***Security Systems:*** Haar cascades can be used in security systems to detect unauthorized access or suspicious activities. They can trigger alarms or notify security personnel when specific objects or behaviours are detected.
6. ***Biometrics:*** Haar cascades can be applied in biometric systems for fingerprint, iris or palm print recognition and among other biometric modalities.
7. ***Image Filtering and Pre processing:*** Haar cascades can be used as a pre-processing step in various image analysis tasks, such as noise reduction, edge detection or feature extraction.

LBPH (Local Binary Patterns Histograms) Algorithm to detect faces. It labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number. LBPH uses 4 parameters such as Radius, Neighbours, Grid X and Grid Y. The model built is trained with the faces with tag given to them, and later on, the machine is given a test data and machine decides the correct label for it.

So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram. We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: **Euclidean distance**, **chi-square**, **absolute value**, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula (i).

$$D = \sqrt{\sum_{i-1}^{n}(hist1_i - hist2_i)^2} \qquad \text{-(i)}$$

D = Euclidean distance

hist1= the coordinate of the first point

hist2= the coordinate of the second point

n= n-space

So the algorithm output is the ID from the image with the closest histogram.

Algorithm:

1. ***Local Binary Patterns (LBP) Computation:***

- For each pixel in the image, the LBP operator examines its neighbourhood (usually a circular region centre at the pixel).
- The LBP operator compares the intensity value of the centre pixel with its surrounding neighbours.
- If a neighbours intensity value is greater than or equal to the centre pixel, it is denoted as "1," otherwise as "0".
- The binary values of all the neighbours are concatenated, forming an LBP code for that particular pixel.

2. ***Histogram Calculation:***

- After computing the LBP code for each pixel, a histogram of LBP codes is constructed for the entire image.
- The histogram bins represent different LBP patterns, and each bin counts the occurrence of its corresponding LBP code in the image.

3. ***Feature Extraction:***

- The final step is feature extraction. The histogram computed in the previous step serves as the feature vector that characterizes the texture or facial features of the image.
- Depending on the application, multiple histograms from different regions of the image can be combined to form a more informative feature vector.

4. ***Classification:***

- Once the feature vectors are extracted for a set of training images, a machine learning algorithm (such as SVM, k-NN or neural networks) can be used for classification.

Advantages:

1. ***Simplicity and efficiency:*** LBPH is a simple and computationally efficient algorithm. It operates directly on gray scale images and does not require complex pre processing steps. Due to its simplicity, it can be implemented easily and runs quickly, making it suitable for real-time applications.
2. ***Robust to illumination changes:*** LBPH is robust to changes in lighting conditions, which is a common challenge in image processing. It achieves this by comparing local pixel neighbourhoods, which helps in capturing texture information independently of global illumination variations.
3. ***Local feature representation:*** LBPH extracts local texture features from an image, which allows it to capture important details in small regions. This makes it effective for tasks where local patterns are essential, such as face recognition, texture classification and object detection.
4. ***Small memory footprint:*** The LBPH algorithm stores only a histogram of binary patterns for each image, making it memory-efficient and suitable for resource-limited devices.

Disadvantages:

1. ***Sensitivity to Image Noise:*** LBPH is sensitive to image noise, which can affect the accuracy of recognition or texture analysis. Noisy images may result in incorrect patterns and lead to false matches or reduced performance.

2. ***Dependency on Image Resolution***: The performance of LBPH can be affected by the resolution of the images. When images have low resolution or are captured from a distance, the local patterns may not be distinct enough for accurate recognition.

3. ***Limited to Local Information:*** LBPH focuses on local patterns within the image. While this can be beneficial for certain applications, it may not capture global or contextual information that could be relevant for some recognition tasks.

4. ***Scalability:*** LBPH can be computationally expensive, especially when dealing with large datasets or real-time applications. The process of extracting local patterns and constructing histograms can be time-consuming.

Applications:

1. ***Face Recognition:*** LBPH is widely used for face recognition tasks. It extracts texture information from facial images and represents them as histograms, making it effective for identifying individuals from a database of faces.

2. ***Object Recognition:*** It can be utilized for recognizing objects in images based on their texture features. For example, it could be employed to identify specific types of fruits or objects with distinctive textures.

3. ***Character Recognition:*** LBPH can be applied to recognize characters or symbols in images, especially in situations where the characters have specific texture patterns.

4. ***Image Retrieval:*** LBPH can be used as a feature descriptor for content-based image retrieval systems. Given a query image, it can help find similar images in a large database based on texture similarities.

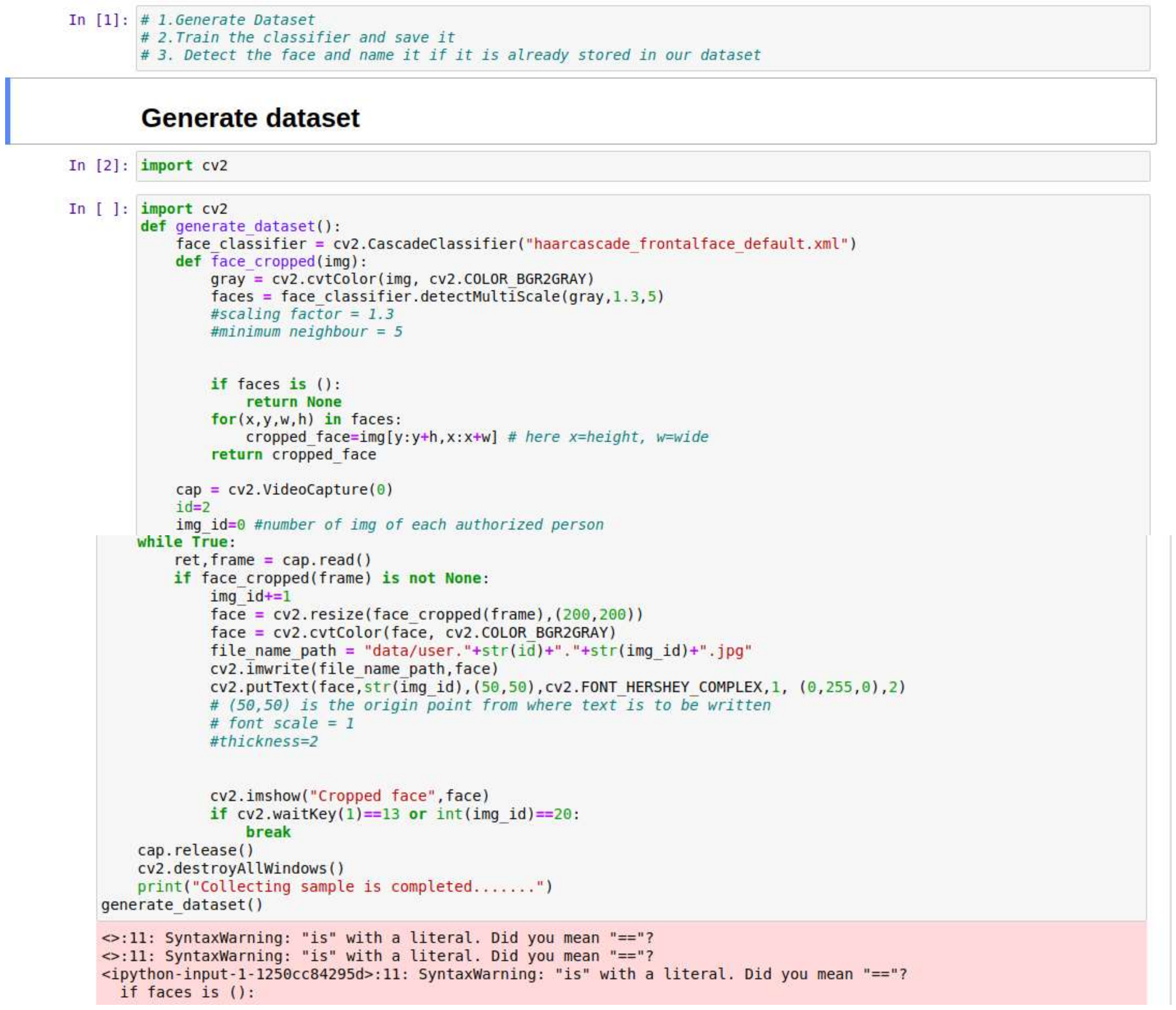
5. ***Gesture Recognition:*** LBPH can be employed in hand gesture recognition applications where texture patterns on the hand are used to identify specific gestures.

6. **Medical Image Analysis:** In medical imaging, LBPH can be used for tasks like tumour detection, tissue classification or identifying anomalies in medical images, based on texture patterns.

7. ***Quality Control in Manufacturing:*** LBPH can be used to inspect the surface texture of products in manufacturing processes and enabling automated quality control.

III. MODULES

MODULE-1: Generate dataset:



```
In [1]: # 1.Generate Dataset
        # 2.Train the classifier and save it
        # 3. Detect the face and name it if it is already stored in our dataset
```

Generate dataset

```
In [2]: import cv2
```

```
In [ ]: import cv2
        def generate_dataset():
            face_classifier = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
            def face_cropped(img):
                gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                faces = face_classifier.detectMultiScale(gray,1.3,5)
                #scaling factor = 1.3
                #minimum neighbour = 5


                if faces is ():
                    return None
                for(x,y,w,h) in faces:
                    cropped_face=img[y:y+h,x:x+w] # here x=height, w=wide
                return cropped_face

            cap = cv2.VideoCapture(0)
            id=2
            img_id=0 #number of img of each authorized person
        while True:
            ret,frame = cap.read()
            if face_cropped(frame) is not None:
                img_id+=1
                face = cv2.resize(face_cropped(frame),(200,200))
                face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
                file_name_path = "data/user."+str(id)+"."+str(img_id)+".jpg"
                cv2.imwrite(file_name_path,face)
                cv2.putText(face,str(img_id),(50,50),cv2.FONT_HERSHEY_COMPLEX,1, (0,255,0),2)
                # (50,50) is the origin point from where text is to be written
                # font scale = 1
                #thickness=2


                cv2.imshow("Cropped face",face)
                if cv2.waitKey(1)==13 or int(img_id)==20:
                    break
        cap.release()
        cv2.destroyAllWindows()
        print("Collecting sample is completed.......")
    generate_dataset()
```

```
<>:11: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:11: SyntaxWarning: "is" with a literal. Did you mean "=="?
<ipython-input-1-1250cc84295d>:11: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if faces is ():
```

Cascading classifier: While collecting open cv python our classifier is automatically installed, so use that in this project.

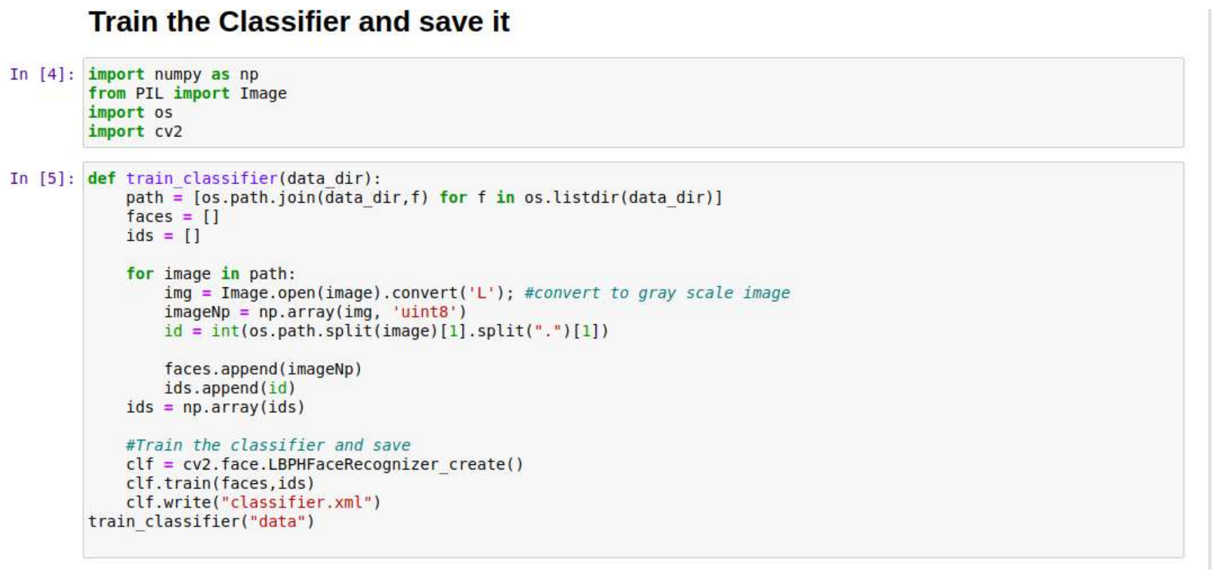
Path in face-classifier:

grey scale image: black and white image.

- ➔ 0 means going to the use the camera from laptop webcam.
- ➔ Id=1 means id of 1st authorized person.
- ➔ 13 means ASCII value of Enter key-> It will go outside of loop [ESC].

- ➔ CV- used for solve computer vision problems.
- ➔ CV2 module is used to build applications that involve image, object detection, recognition, and tracking.
- ➔ Using haarcascade_fronta1face_defaut. Xml to get the co-ordinate of face so that, It can crop and use it to make a dataset and for future prediction.
- ➔ Using gray scale image means RGB image has 3 channels whereas greyscale image has only 1 channel. It reduces a lot of complexity if it converts.
- ➔ In this project use face classifier and using detect multi scale method, get the coordinates (x, y).

Train the classifier and save it:

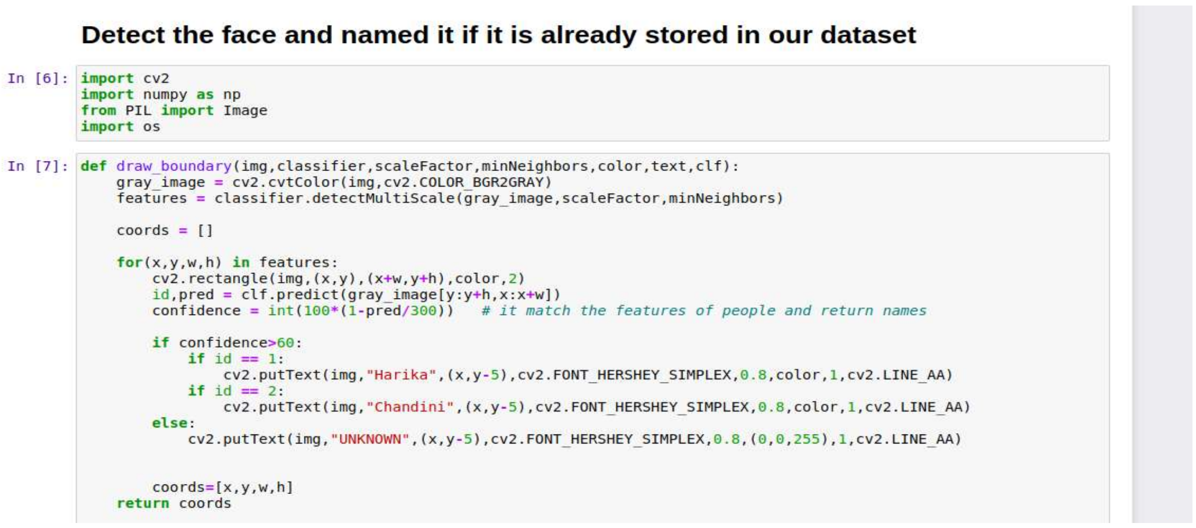

Train the Classifier and save it

```
In [4]: import numpy as np
        from PIL import Image
        import os
        import cv2
```

```
In [5]: def train_classifier(data_dir):
            path = [os.path.join(data_dir,f) for f in os.listdir(data_dir)]
            faces = []
            ids = []

            for image in path:
                img = Image.open(image).convert('L'); #convert to gray scale image
                imageNp = np.array(img, 'uint8')
                id = int(os.path.split(image)[1].split(".")[1])

                faces.append(imageNp)
                ids.append(id)
            ids = np.array(ids)

            #Train the classifier and save
            clf = cv2.face.LBPHFaceRecognizer_create()
            clf.train(faces,ids)
            clf.write("classifier.xml")
        train_classifier("data")
```

- ➔ In this classifier.xml because our system doesn't understand images. It needs to be converted into a Numpy array format.
- ➔ So that it can make certain decisions based on this data.
- ➔ Classifer.xml includes certain information (data) about faces.

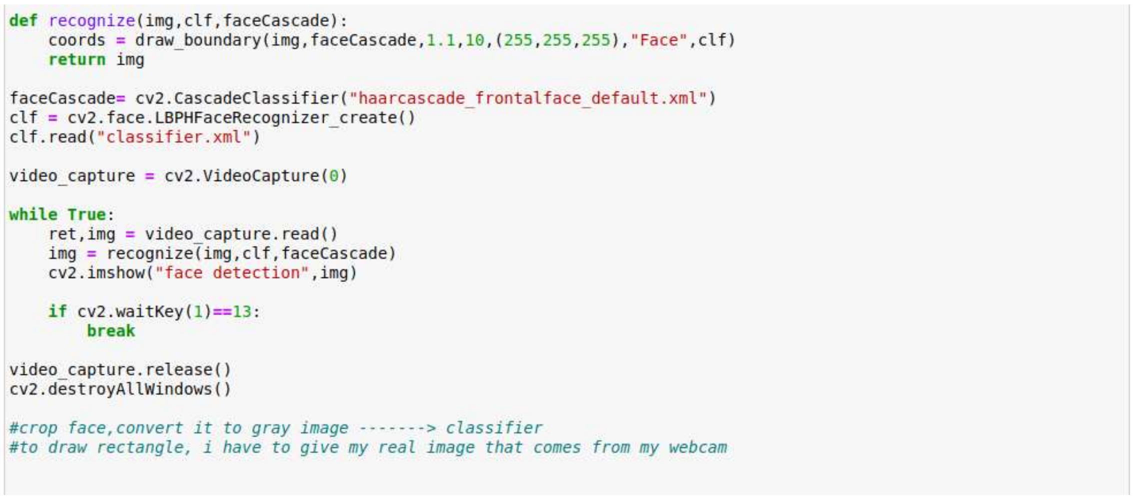
MODULE-2: Detect the face and named it if it is already stored in our dataset.



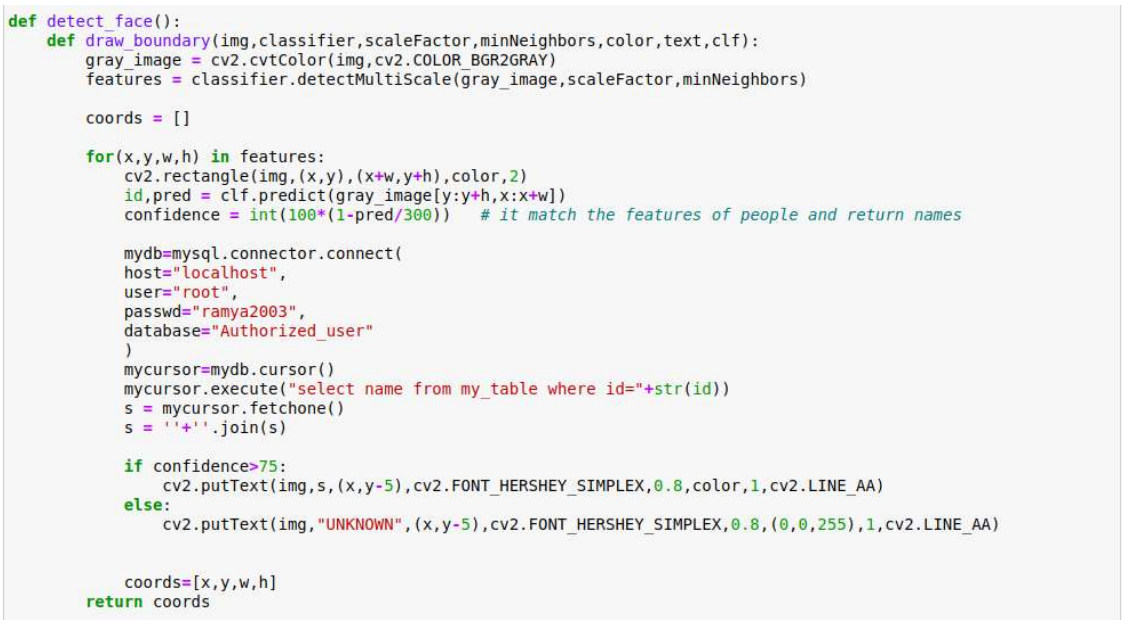
Detect the face and named it if it is already stored in our dataset

```
In [6]: import cv2
        import numpy as np
        from PIL import Image
        import os
```

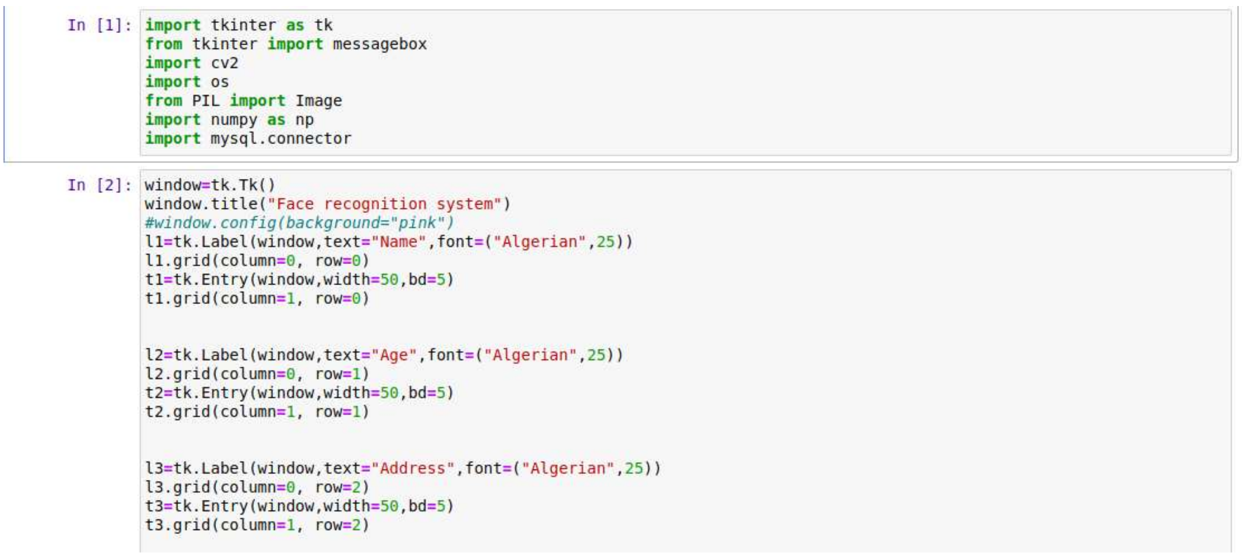
```
In [7]: def draw_boundary(img,classifier,scaleFactor,minNeighbors,color,text,clf):
            gray_image = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            features = classifier.detectMultiScale(gray_image,scaleFactor,minNeighbors)

            coords = []

            for(x,y,w,h) in features:
                cv2.rectangle(img,(x,y),(x+w,y+h),color,2)
                id,pred = clf.predict(gray_image[y:y+h,x:x+w])
                confidence = int(100*(1-pred/300))   # it match the features of people and return names

                if confidence>60:
                    if id == 1:
                        cv2.putText(img,"Harika",(x,y-5),cv2.FONT_HERSHEY_SIMPLEX,0.8,color,1,cv2.LINE_AA)
                    if id == 2:
                        cv2.putText(img,"Chandini",(x,y-5),cv2.FONT_HERSHEY_SIMPLEX,0.8,color,1,cv2.LINE_AA)
                else:
                    cv2.putText(img,"UNKNOWN",(x,y-5),cv2.FONT_HERSHEY_SIMPLEX,0.8,(0,0,255),1,cv2.LINE_AA)


                coords=[x,y,w,h]
            return coords
```



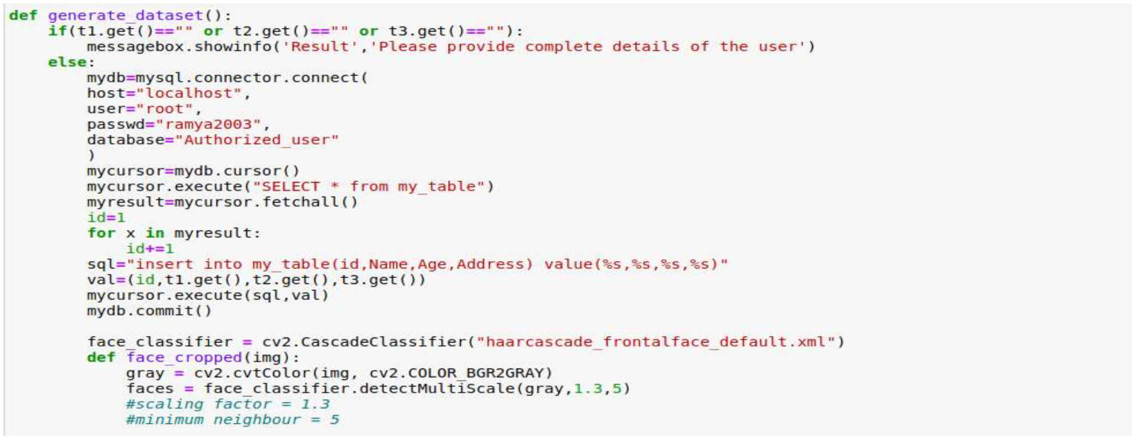
```
def recognize(img,clf,faceCascade):
    coords = draw_boundary(img,faceCascade,1.1,10,(255,255,255),"Face",clf)
    return img

faceCascade= cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
clf = cv2.face.LBPHFaceRecognizer_create()
clf.read("classifier.xml")

video_capture = cv2.VideoCapture(0)

while True:
    ret,img = video_capture.read()
    img = recognize(img,clf,faceCascade)
    cv2.imshow("face detection",img)

    if cv2.waitKey(1)==13:
        break

video_capture.release()
cv2.destroyAllWindows()

#crop face,convert it to gray image -------> classifier
#to draw rectangle, i have to give my real image that comes from my webcam
```



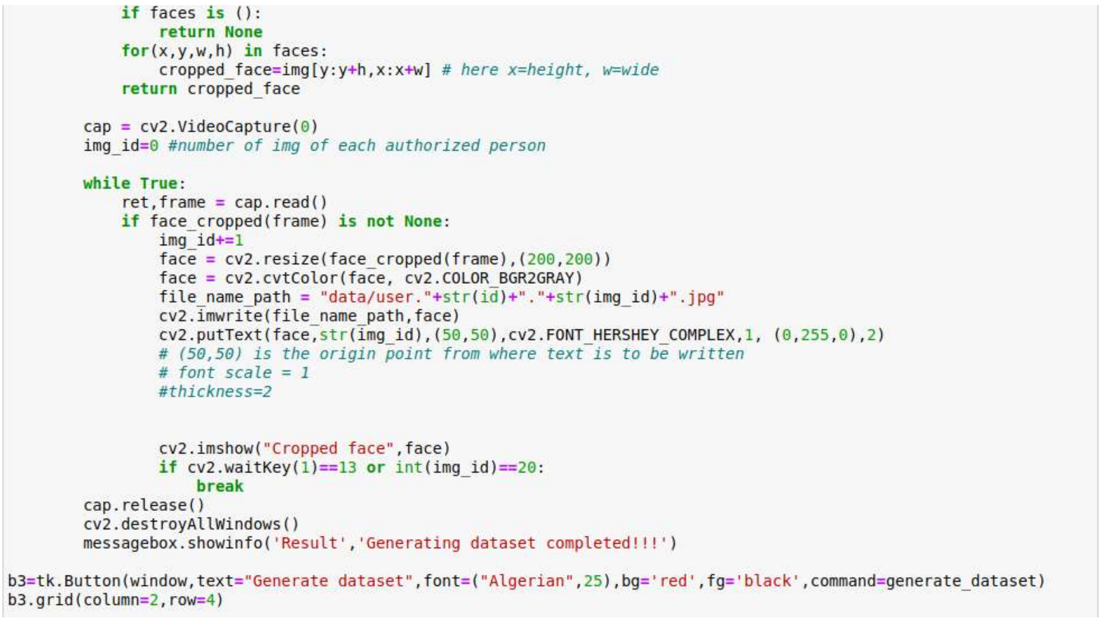
```
def detect_face():
    def draw_boundary(img,classifier,scaleFactor,minNeighbors,color,text,clf):
        gray_image = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        features = classifier.detectMultiScale(gray_image,scaleFactor,minNeighbors)

        coords = []

        for(x,y,w,h) in features:
            cv2.rectangle(img,(x,y),(x+w,y+h),color,2)
            id,pred = clf.predict(gray_image[y:y+h,x:x+w])
            confidence = int(100*(1-pred/300))    # it match the features of people and return names

            mydb=mysql.connector.connect(
            host="localhost",
            user="root",
            passwd="ramya2003",
            database="Authorized_user"
            )
            mycursor=mydb.cursor()
            mycursor.execute("select name from my_table where id="+str(id))
            s = mycursor.fetchone()
            s = ''+''.join(s)

            if confidence>75:
                cv2.putText(img,s,(x,y-5),cv2.FONT_HERSHEY_SIMPLEX,0.8,color,1,cv2.LINE_AA)
            else:
                cv2.putText(img,"UNKNOWN",(x,y-5),cv2.FONT_HERSHEY_SIMPLEX,0.8,(0,0,255),1,cv2.LINE_AA)


            coords=[x,y,w,h]
        return coords
```



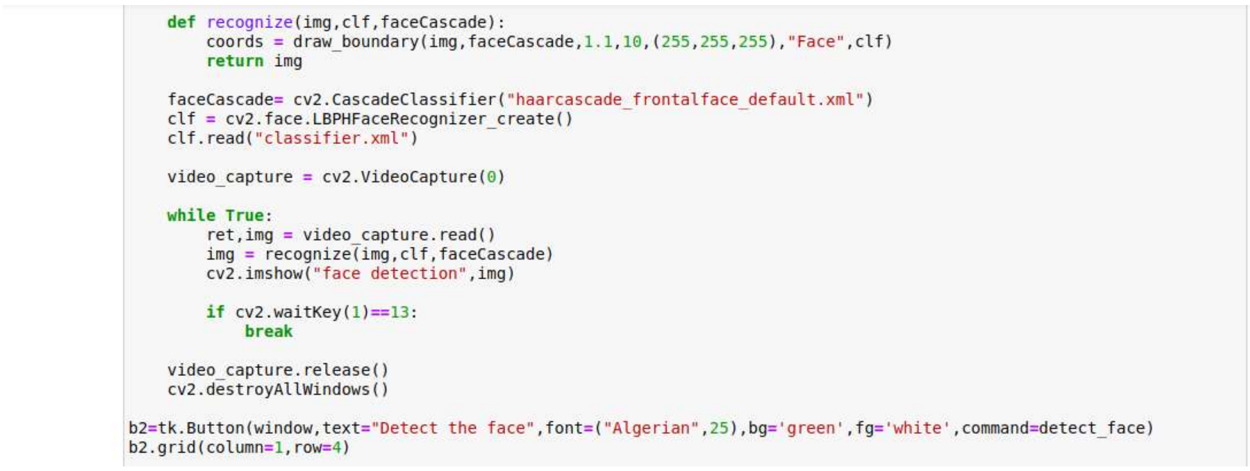
```
In [1]: import tkinter as tk
        from tkinter import messagebox
        import cv2
        import os
        from PIL import Image
        import numpy as np
        import mysql.connector

In [2]: window=tk.Tk()
        window.title("Face recognition system")
        #window.config(background="pink")
        l1=tk.Label(window,text="Name",font=("Algerian",25))
        l1.grid(column=0, row=0)
        t1=tk.Entry(window,width=50,bd=5)
        t1.grid(column=1, row=0)


        l2=tk.Label(window,text="Age",font=("Algerian",25))
        l2.grid(column=0, row=1)
        t2=tk.Entry(window,width=50,bd=5)
        t2.grid(column=1, row=1)


        l3=tk.Label(window,text="Address",font=("Algerian",25))
        l3.grid(column=0, row=2)
        t3=tk.Entry(window,width=50,bd=5)
        t3.grid(column=1, row=2)
```



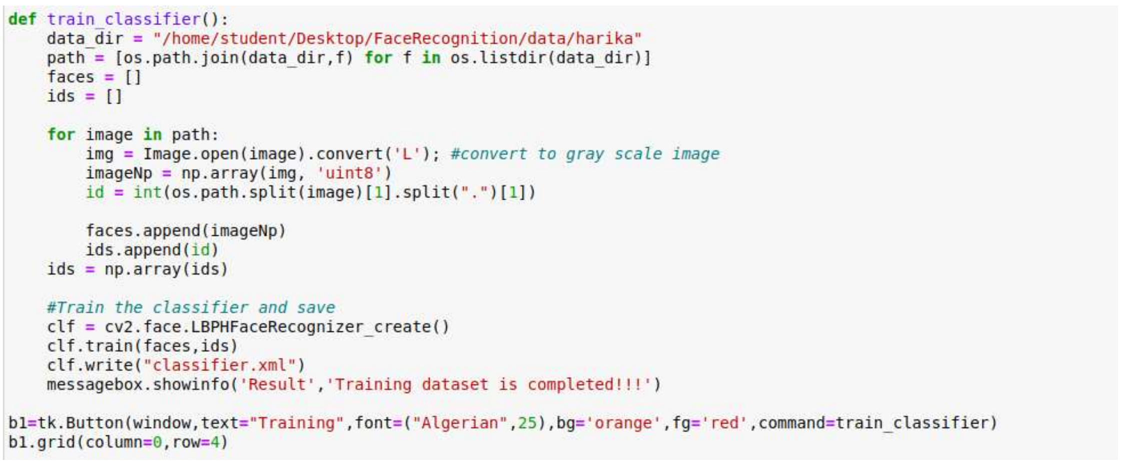
```python
def generate_dataset():
    if(t1.get()=="" or t2.get()=="" or t3.get()==""):
        messagebox.showinfo('Result','Please provide complete details of the user')
    else:
        mydb=mysql.connector.connect(
        host="localhost",
        user="root",
        passwd="ramya2003",
        database="Authorized_user"
        )
        mycursor=mydb.cursor()
        mycursor.execute("SELECT * from my_table")
        myresult=mycursor.fetchall()
        id=1
        for x in myresult:
            id+=1
        sql="insert into my_table(id,Name,Age,Address) value(%s,%s,%s,%s)"
        val=(id,t1.get(),t2.get(),t3.get())
        mycursor.execute(sql,val)
        mydb.commit()

        face_classifier = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
        def face_cropped(img):
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = face_classifier.detectMultiScale(gray,1.3,5)
            #scaling factor = 1.3
            #minimum neighbour = 5
```



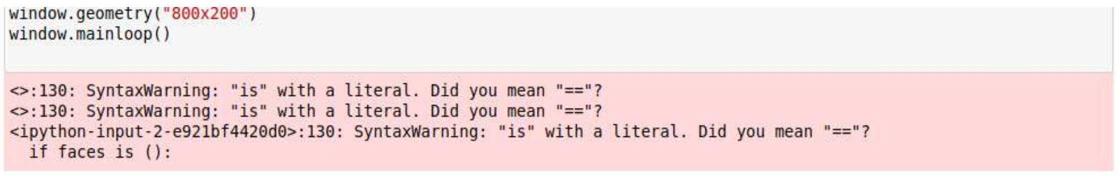
```python
            if faces is ():
                return None
            for(x,y,w,h) in faces:
                cropped_face=img[y:y+h,x:x+w] # here x=height, w=wide
            return cropped_face

        cap = cv2.VideoCapture(0)
        img_id=0 #number of img of each authorized person

        while True:
            ret,frame = cap.read()
            if face_cropped(frame) is not None:
                img_id+=1
                face = cv2.resize(face_cropped(frame),(200,200))
                face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
                file_name_path = "data/user."+str(id)+"."+str(img_id)+".jpg"
                cv2.imwrite(file_name_path,face)
                cv2.putText(face,str(img_id),(50,50),cv2.FONT_HERSHEY_COMPLEX,1, (0,255,0),2)
                # (50,50) is the origin point from where text is to be written
                # font scale = 1
                #thickness=2


                cv2.imshow("Cropped face",face)
                if cv2.waitKey(1)==13 or int(img_id)==20:
                    break
        cap.release()
        cv2.destroyAllWindows()
        messagebox.showinfo('Result','Generating dataset completed!!!')

b3=tk.Button(window,text="Generate dataset",font=("Algerian",25),bg='red',fg='black',command=generate_dataset)
b3.grid(column=2,row=4)
```



```
    def recognize(img,clf,faceCascade):
        coords = draw_boundary(img,faceCascade,1.1,10,(255,255,255),"Face",clf)
        return img

    faceCascade= cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
    clf = cv2.face.LBPHFaceRecognizer_create()
    clf.read("classifier.xml")

    video_capture = cv2.VideoCapture(0)

    while True:
        ret,img = video_capture.read()
        img = recognize(img,clf,faceCascade)
        cv2.imshow("face detection",img)

        if cv2.waitKey(1)==13:
            break

    video_capture.release()
    cv2.destroyAllWindows()

b2=tk.Button(window,text="Detect the face",font=("Algerian",25),bg='green',fg='white',command=detect_face)
b2.grid(column=1,row=4)
```



```
def train_classifier():
    data_dir = "/home/student/Desktop/FaceRecognition/data/harika"
    path = [os.path.join(data_dir,f) for f in os.listdir(data_dir)]
    faces = []
    ids = []

    for image in path:
        img = Image.open(image).convert('L'); #convert to gray scale image
        imageNp = np.array(img, 'uint8')
        id = int(os.path.split(image)[1].split(".")[1])

        faces.append(imageNp)
        ids.append(id)
    ids = np.array(ids)

    #Train the classifier and save
    clf = cv2.face.LBPHFaceRecognizer_create()
    clf.train(faces,ids)
    clf.write("classifier.xml")
    messagebox.showinfo('Result','Training dataset is completed!!!')

b1=tk.Button(window,text="Training",font=("Algerian",25),bg='orange',fg='red',command=train_classifier)
b1.grid(column=0,row=4)
```



```
window.geometry("800x200")
window.mainloop()
```

```
<>:130: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:130: SyntaxWarning: "is" with a literal. Did you mean "=="?
<ipython-input-2-e921bf4420d0>:130: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if faces is ():
```

- ➔ Here using draw boundary function, which draw a boundary around the face and also display the name at the top of the face?
- ➔ In this module import image from PIL (python imaging library). Instead of PILLOW module we use PIL.
- ➔ This module is used to adds support for opening, manipulating, and saving many different image file formats

Up to here we use 2 algorithms:

1.Haar Cascade:

That can detect objects in images, irrespective of their scale in image and location. This algorithm is not so complex and can run in real-time.

It can train a harr-cascade detector to detect various faces.

2.LBPH algorithm:

- ➔ LBPH (Local Binary Pattern Histograms). Algorithm to detect faces. It labels the pixels of an image by their holding the neighbourhood of each pixel and considers the results as a binary number.
- ➔ LBPH consists (or) uses 4 parameters:

  i. ***Radius:*** The radius is used to build the circular local binary pattern and represents the radius the radius around the central pixel.

  ii. ***Neighbours:*** The no. Of sample points to build the circular local binary pattern.

  iii. ***Grid x:*** The no. of cells in the horizontal direction.

  iv. ***Grid y:*** The no. of cells in the vertical direction.

➔ The model built it trained with the faces with tag given to them, and later on the machine is given a test data and machine decides the correct label for it.

Converting the project into GUI:

- ➔ Convert the project into GUI means it is very important to convert any of our python projects in GUI because if we want to submit our project to our college or any other places it is very good to have our project in the GUI application.
- ➔ It is because, it can perform every operation within the same window.
- ➔ Don’t need to execute each cell of code again and again.
- ➔ tkinter is a module used to display the window.
- ➔ From tkinter we import message box.
- ➔ tk.button() function is used to create the button.
- ➔ tk.label() function is used to create a label.
- ➔ window.geometry() function is used to represent the size of the window.
- ➔ window.title() function is used to represent the title.
- ➔ tk.Entry() function is used to create the text area to enter details.
- ➔ Window.mainloop() function tells python to run the tkinter event loop.

MODULE-3: Connect to database.

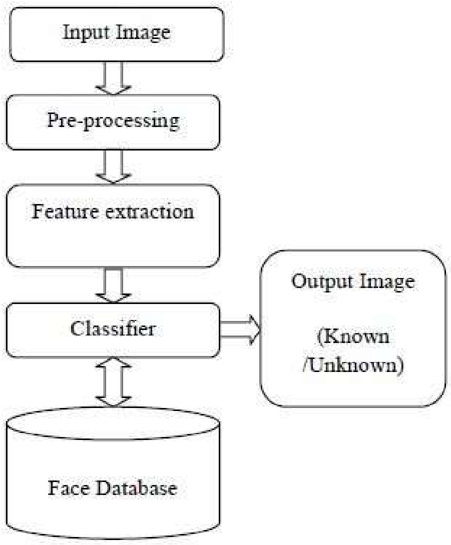
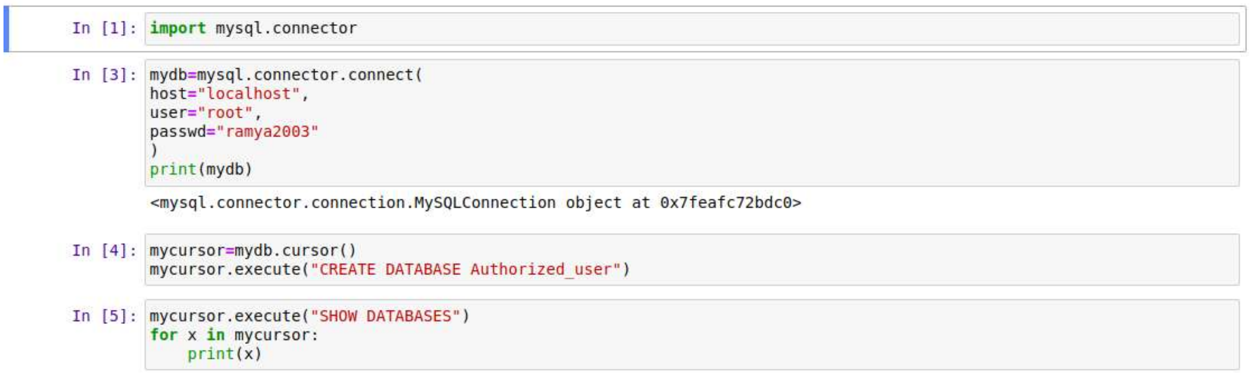
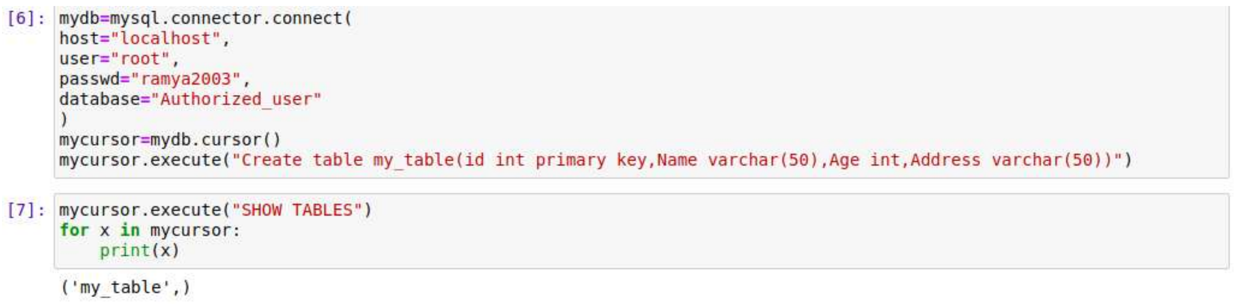


Figure 3.1: Flowchart of face recognition

As show in figure 3.1, the face recognition system takes three input parameters such as name, age and address and by click on the generate dataset button it will generating the dataset. Those input saved in database.



```
In [1]: import mysql.connector

In [3]: mydb=mysql.connector.connect(
        host="localhost",
        user="root",
        passwd="ramya2003"
        )
        print(mydb)

        <mysql.connector.connection.MySQLConnection object at 0x7feafc72bdc0>

In [4]: mycursor=mydb.cursor()
        mycursor.execute("CREATE DATABASE Authorized_user")

In [5]: mycursor.execute("SHOW DATABASES")
        for x in mycursor:
            print(x)
```



```
[6]: mydb=mysql.connector.connect(
     host="localhost",
     user="root",
     passwd="ramya2003",
     database="Authorized_user"
     )
     mycursor=mydb.cursor()
     mycursor.execute("Create table my_table(id int primary key,Name varchar(50),Age int,Address varchar(50))")

[7]: mycursor.execute("SHOW TABLES")
     for x in mycursor:
         print(x)

     ('my_table',)
```

The face recognition system follows the steps as show in figure 3.1. The functionalities of face recognition modules works as follows

- ➔ Here store the details of the persons by using the database.
- ➔ For this import the module MYSQL.CONNECTOR by using this we connect to database.
- ➔ The name of the database is Authorized_user.
- ➔ The table name use in this database is my_table.
- ➔ Use the cusor() function to allow python to execute sql commands in a database session.
- ➔ Use Fetchall() method to retrieves all the rows in the result set of a query returns them list of tuples.

Converting the project into .exe file (app):

- ➔ First install pyinstaller.
- ➔ Next run the command **“pyinstaller –onefile -wGUI_Face_Recognition.py ”**

- ➔ By this command we convert GUI_Face_Recognition.py into .exe file (app).

IV. RESULTS AND DISCUSSION

As show in the below figures, convert GUI_Face_Recognition.py into exe file (app). generating dataset and training dataset is completed it detect the faces and name.

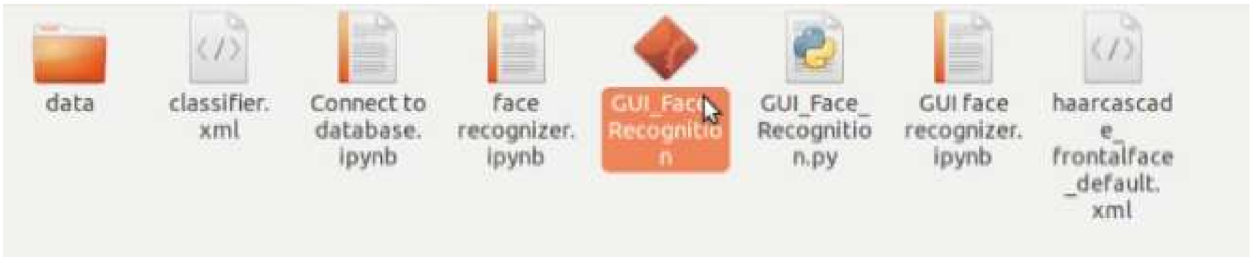

data
classifier.xml
Connect to database.ipynb
face recognizer.ipynb
GUI_Face_Recognition
GUI_Face_Recognition.py
GUI face recognizer.ipynb
haarcascade_frontalface_default.xml

Figure 4.1: GUI_Face_Recognition.py into .exe file (app).

As show in figure 4.1, we convert GUI_Face_Recognition.py into .exe file (app).

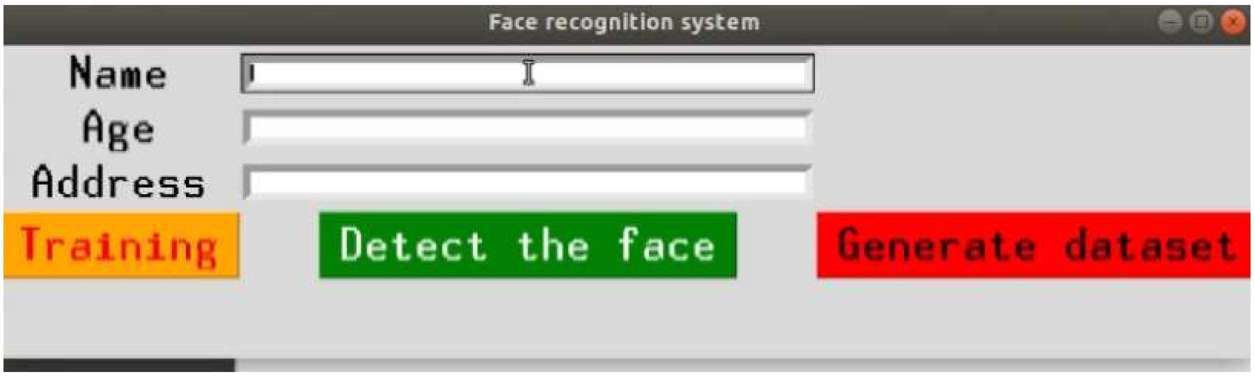
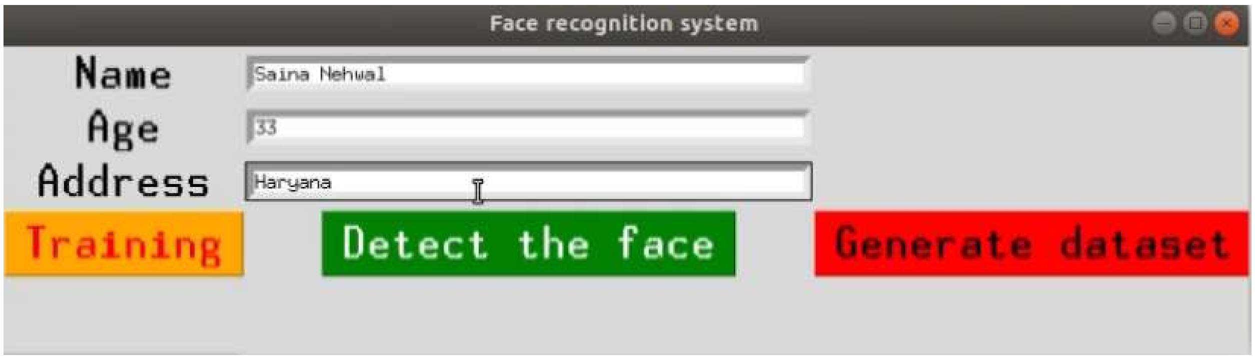


Figure 4.2: By run the app this window will open

As show in figure 4.2, the face recognition systems takes three input parameters such as name, age, address and also the buttons like Training and Detect the face and generate dataset.

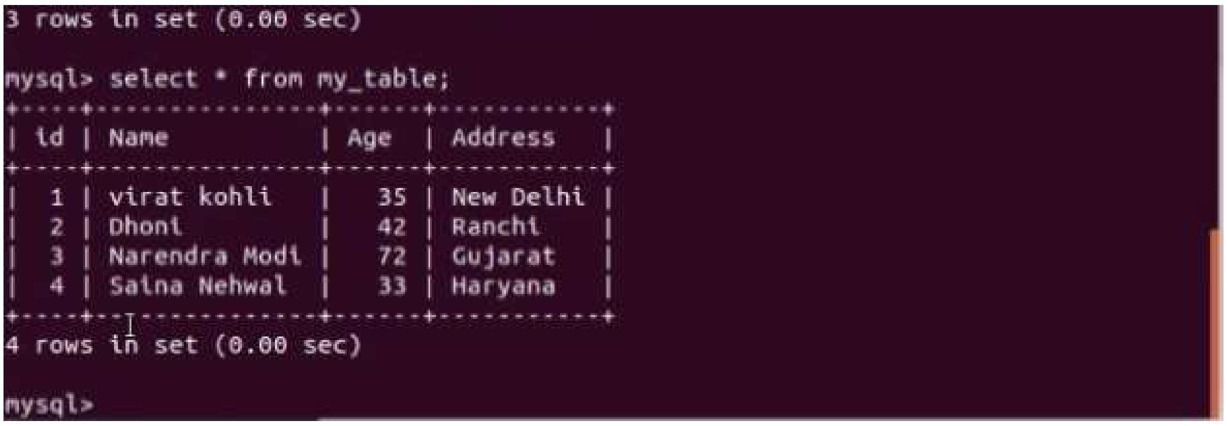
- **Training:** It improves the algorithm's ability to determine whether there are faces in an image.
- **Detect the face:** It identifies human faces in digital images.
- **Generate dataset:** By taking the given inputs it will generate the dataset.



Name: Saina Nehwal
Age: 33
Address: Haryana

Figure 4.3: Entering the inputs

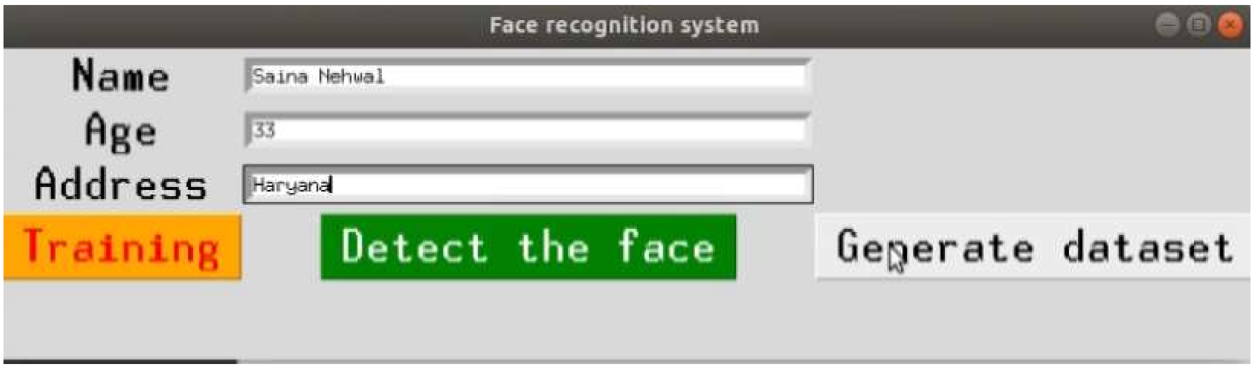
As show in figure 4.3, the face recognition system takes three input parameters such as Saina Nehwal, 33, Haryana and also the buttons like Training, Detect the face and generate dataset.



```
3 rows in set (0.00 sec)

mysql> select * from my_table;
+----+----------------+------+------------+
| id | Name           | Age  | Address    |
+----+----------------+------+------------+
|  1 | virat kohli    |   35 | New Delhi  |
|  2 | Dhoni          |   42 | Ranchi     |
|  3 | Narendra Modi  |   72 | Gujarat    |
|  4 | Saina Nehwal   |   33 | Haryana    |
+----+----------------+------+------------+
4 rows in set (0.00 sec)

mysql>
```

Figure 4.4: Entered data is stored in database

As show in figure 4.4, in database id, name, age, address are stored in the table format by the table name.



Saina Nehwal

33

Haryana

Figure 4.5: Generating the dataset

As show in figure 4.5, the face recognition system takes three input parameters such as name, age, address and by click on the generate dataset button it will generating the dataset.

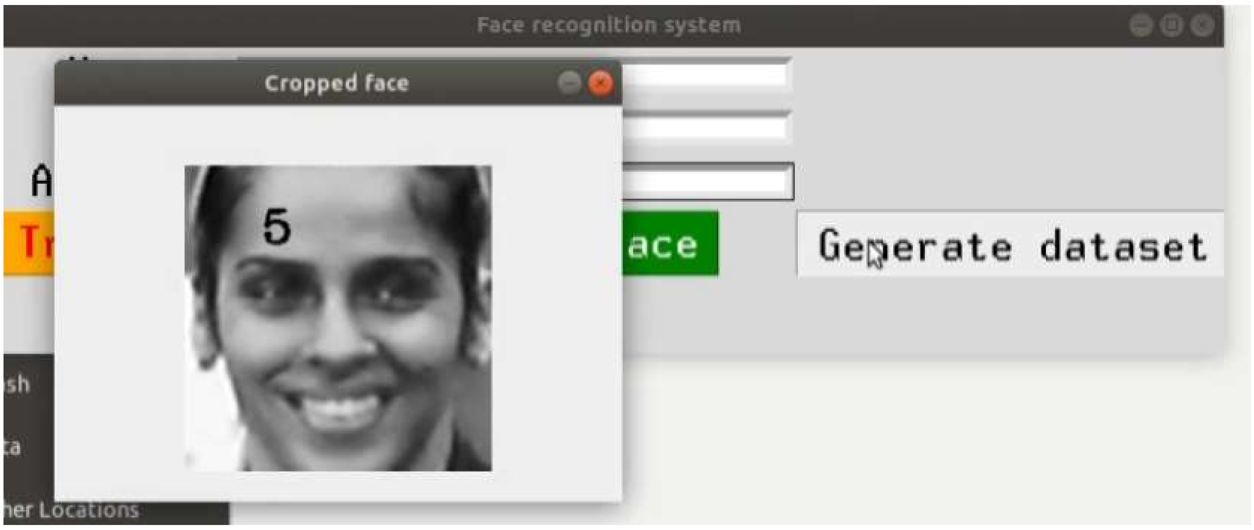
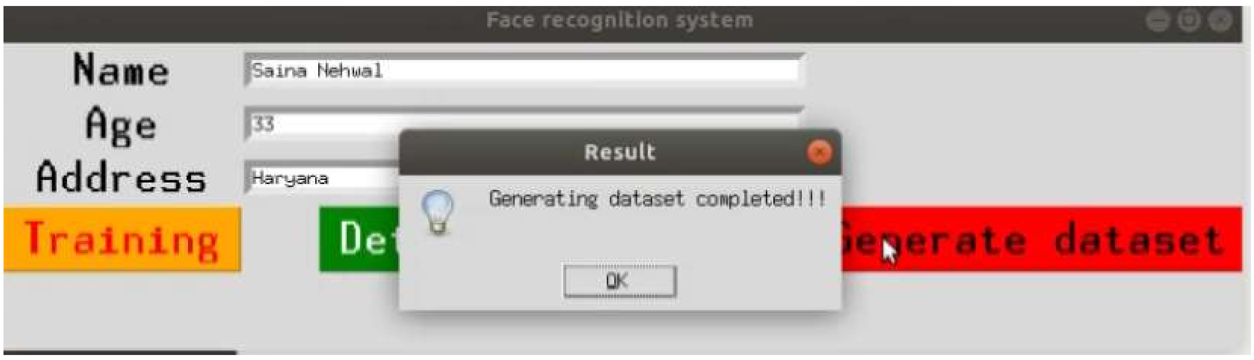


Figure 4.6: Here dataset is generating

As show in figure 4.6, the face recognition system takes three input parameters such as name, age, address and by click on the generate dataset button it will generating the dataset.



Saina Nehwal

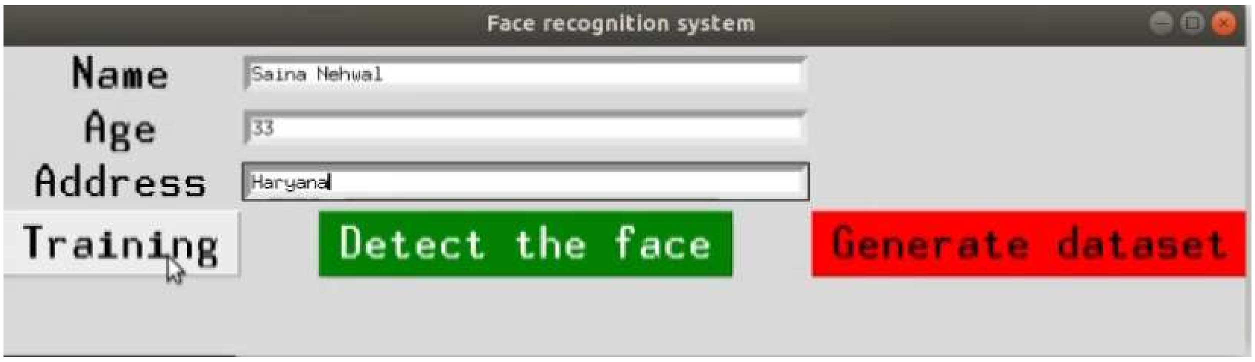
33

Haryana

Generating dataset completed!!!

Figure 4.7: Generating dataset is completed

As show in figure 4.7, the face recognition system takes three input parameters such as name, age, address and by click on the generate dataset button it will generating the dataset and generating dataset is completed.



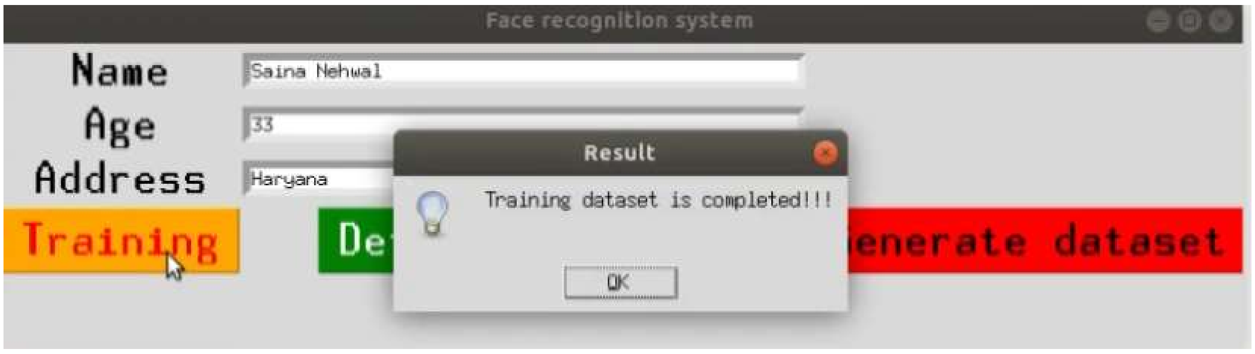
Saina Nehwal

33

Haryana

Figure 4.8: Training the dataset

As show in figure 4.8, the face recognition system takes three input parameters such as name, age, address and after generating dataset is completed. By click on the Training button it will training the dataset.



Saina Nehwal

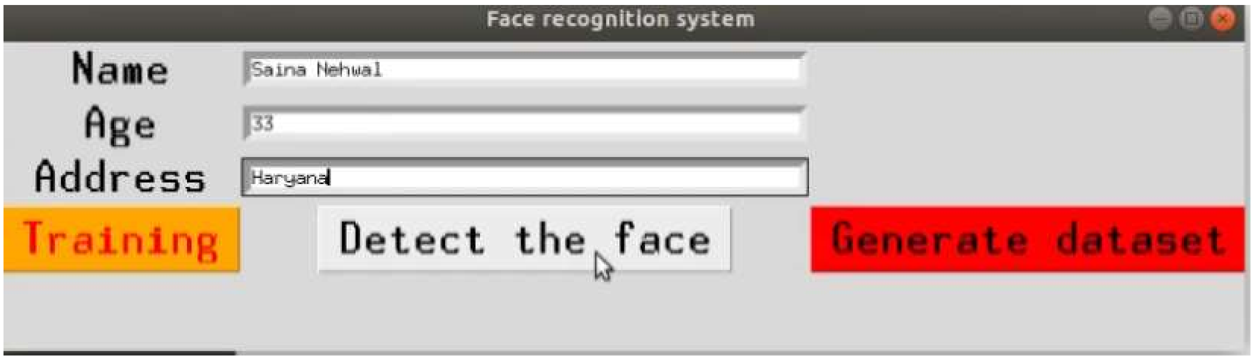
33

Haryana

Training dataset is completed!!!

Figure 4.9: Training dataset is completed

As show in figure 4.9, the face recognition system takes three input parameters such as name, age, address and after generating dataset is completed. By click on the Training button it will training the dataset and training dataset is completed.



Saina Nehwal

33

Haryana

Figure 4.10: Detect the face

As show in figure 4.10, the face recognition system takes three input parameters such as name, age, address and after generating dataset and training dataset is completed. Click on the detect the face button.

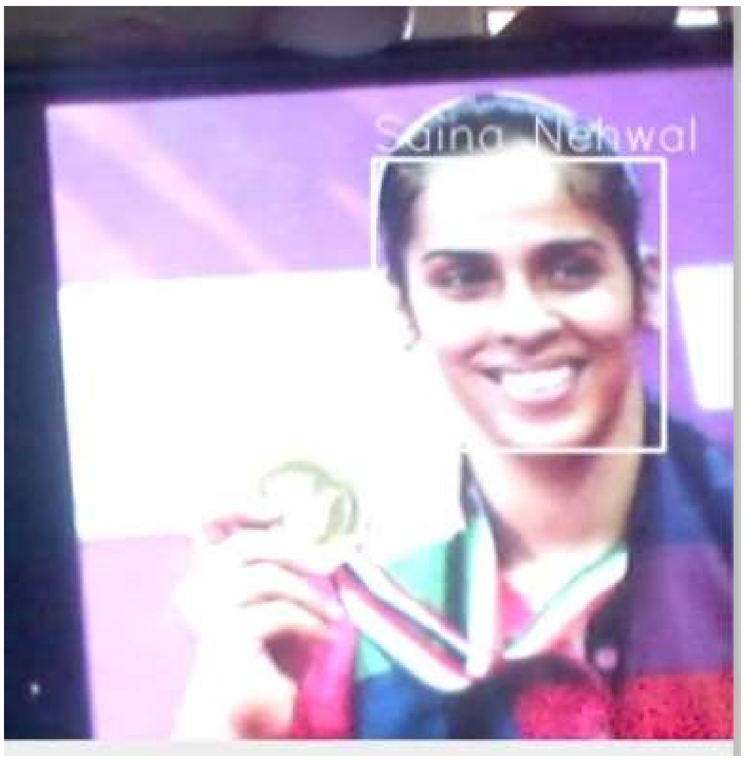


Figure 4.11: Here face is detected

As show in figure 4.11, the face recognition system takes three input parameters such as name, age, address and after generating dataset and training dataset is completed it detect the faces and name.

V. CONCLUSION AND FUTURE ENHANCEMENTS

Conclusion: Finally create the face recognition app which is used to generate dataset, train classifier and detect the faces. It is used to recognize the faces of the people which are trained in the dataset.

Future Enhancements:

1. ***Improved Accuracy and Robustness:*** Despite the progress, face recognition systems can still face challenges in real-world scenarios, such as low-resolution images, occlusions, and disguise attempts. Future research should focus on enhancing the accuracy and robustness of the systems, ensuring reliable performance under diverse conditions.
2. ***Ethical and Privacy Considerations:*** As face recognition technology becomes more prevalent, ethical concerns and privacy issues arise. Future enhancements should prioritize privacy protection and ensure that face recognition systems are used responsibly and with proper consent from individuals.
3. ***Cross-Dataset Generalization:*** Face recognition models often struggle to generalize well across different datasets. Future research should work towards developing models that can perform consistently and effectively on diverse datasets, reducing the need for extensive retraining for specific scenarios.
4. ***Real-Time Performance:*** While significant progress has been made, real-time face recognition on large-scale databases is still challenging. Future improvements should focus on optimizing model architectures and algorithms to achieve real-time performance without sacrificing accuracy.
5. ***Multimodal Approaches:*** Integrating multiple biometric modalities, such as face and voice recognition, could enhance the overall performance and security of authentication systems.
6. ***Cross-Domain Applications:*** Face recognition system has great potential in various fields beyond security and authentication. Future enhancements should explore cross-domain applications, such as healthcare, marketing, and human-computer interaction.

face recognition has come a long way and holds great promise for the furture. With ongoing research and development, the system will become more accurate, reliable, and ethical, enabling a wide range of applications that benefit society while addressing potential challenges and concerns.

REFERENCES

. [1] S. Bhattacharya, G. S. Nainala, P. Das and A.Routray, “Smart Attendance System (SAS): A Face Recognition Based Attendance System for Classroom Environment”, in 2018.

[2] J. Deng, Jia Guo, Niannan X, Irene, Stefanos, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition", Journal of LATEX Class Files, Vol. 14, No. 8, August 2019.

[3] Smitha, Pavithra S Hegde, Afshin, “Face Recognition based Attendance Management System”, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181 Vol. 9, Issue 05, May-2020.

[4] S. Qu, W. Xu, J. Zhao, and H. Zhang, Lee et al, “Design and implementation of a fast sliding-mode speed controller with disturbance compensation for spasm system” , *on* vol. 99, 2021.

[5] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Yu Qiao, Wang et al, “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks”, IEEE SIGNAL PROCESSING LETTERS, VOL. 23, NO. 10, OCTOBER 2022.

[6] Yaniv T, Ming Y, Marc AR, Lior W, Garcia et al, "Deep Face: Closing the Gap to Human-Level Performance in Face Verification”, available at: https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf