# ADVANCE ATTANDANCE SYSTEM [AAS]

## [USING MACHINE LEARNING AND DEEP LEARNING]

# ADVANCE ATTANDANCE SYSTEM

BY:
**SAGAR NAVAL,ABHISHEK SAIN**
Under the guidance of
**Mr. PURUSHOTAM SHARMA**
**TRAINER, MACHINE LEARNING AND DEEP LEARNING**
Department of Data Analytics and Machine Learning
Imarticus Learning

# CONTENTS

INTRODUCTION

METHODOLOGY

DATA COLLECTION &PREPROCESSING

CREATING FILE FOR AAS

CLUSTERING OF DATA

RECORD OF ATTANDANCE

CONCLUSION

# INTRODUCTION
## BY OPENCV & KNN ABSTRACT

1. The Advanced Attendance System is an automated solution designed for efficient attendance tracking.

2. It incorporates modern technologies, including facial recognition, computer vision, and real-time processing.

3. By utilizing facial recognition, the system identifies individuals and creates digital representations of their unique facial features.

4. Real-time processing enables instant attendance recording as individuals enter or leave the designated area.

5. The system generates comprehensive attendance reports automatically, providing valuable insights for administrators and supervisors.

6. Enhanced security measures prevent unauthorized access and eliminate the possibility of proxy attendance.

7. The Advanced Attendance System saves time, reduces administrative burden, making it suitable for educational institutions, organizations, and events.

# METHODOLOGY

- **The project leverages OpenCV (cv2) for image and video data collection and employs K-Nearest Neighbors (KNN) as an unsupervised machine learning algorithm for facial recognition in an advanced attendance system.**

- **OpenCV (cv2) is an open-source computer vision and machine learning library used for image and video processing tasks.**

- **It provides functions for reading, writing, and manipulating images/videos, as well as object detection and camera calibration.**

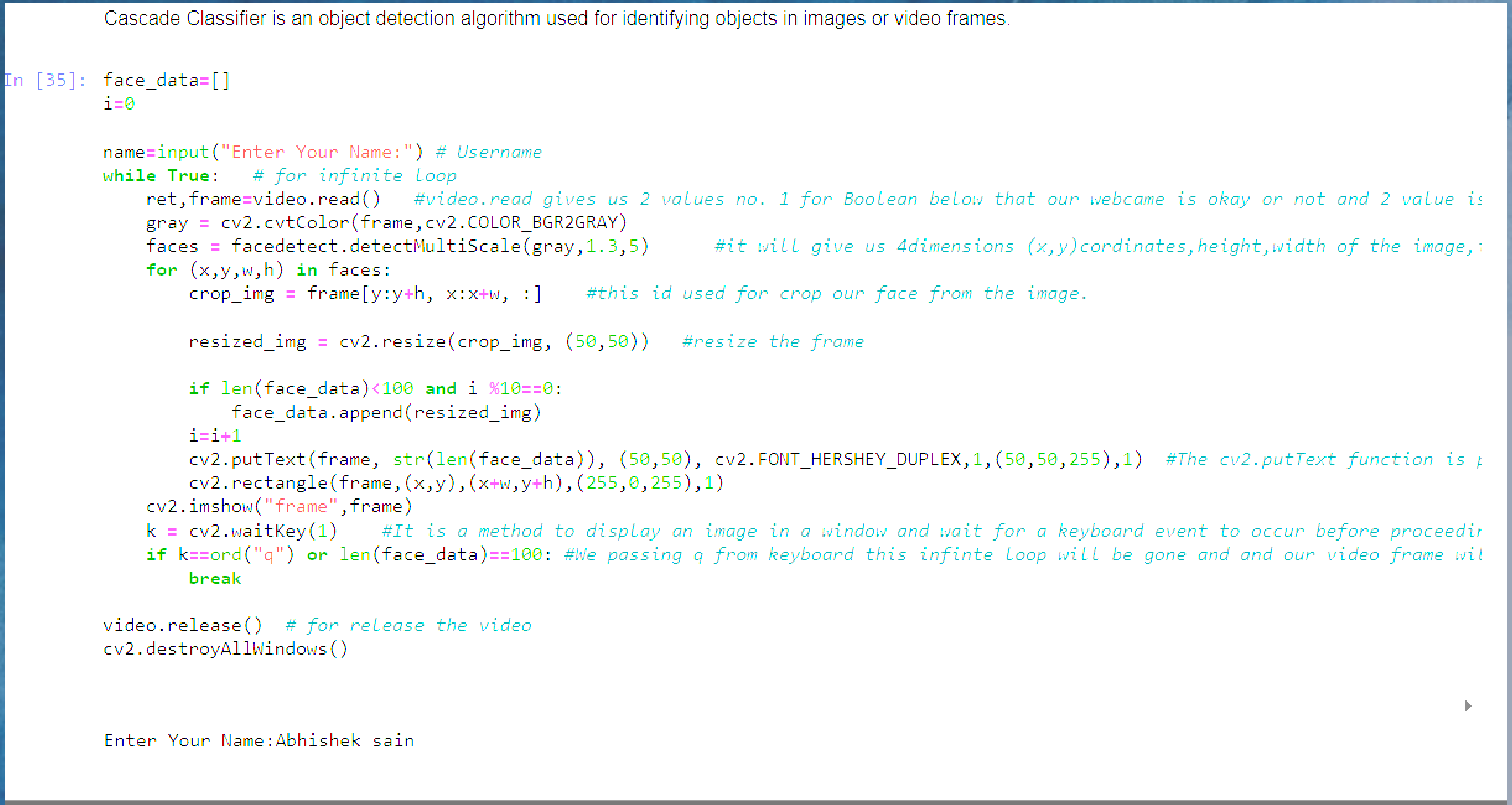
- It offers a vast collection of pre-built functions for tasks like image filtering, feature extraction, and geometric transformations.

- **K-Nearest Neighbors (KNN) is a simple and non-parametric machine learning algorithm used for classification.**

- **It makes predictions based on the majority class of the 'K' nearest data points to the new input.**

- **KNN is a non-parametric algorithm, meaning it doesn't make assumptions about the underlying data distribution.**

# DATA COLLECTION & PREPROCESSING:



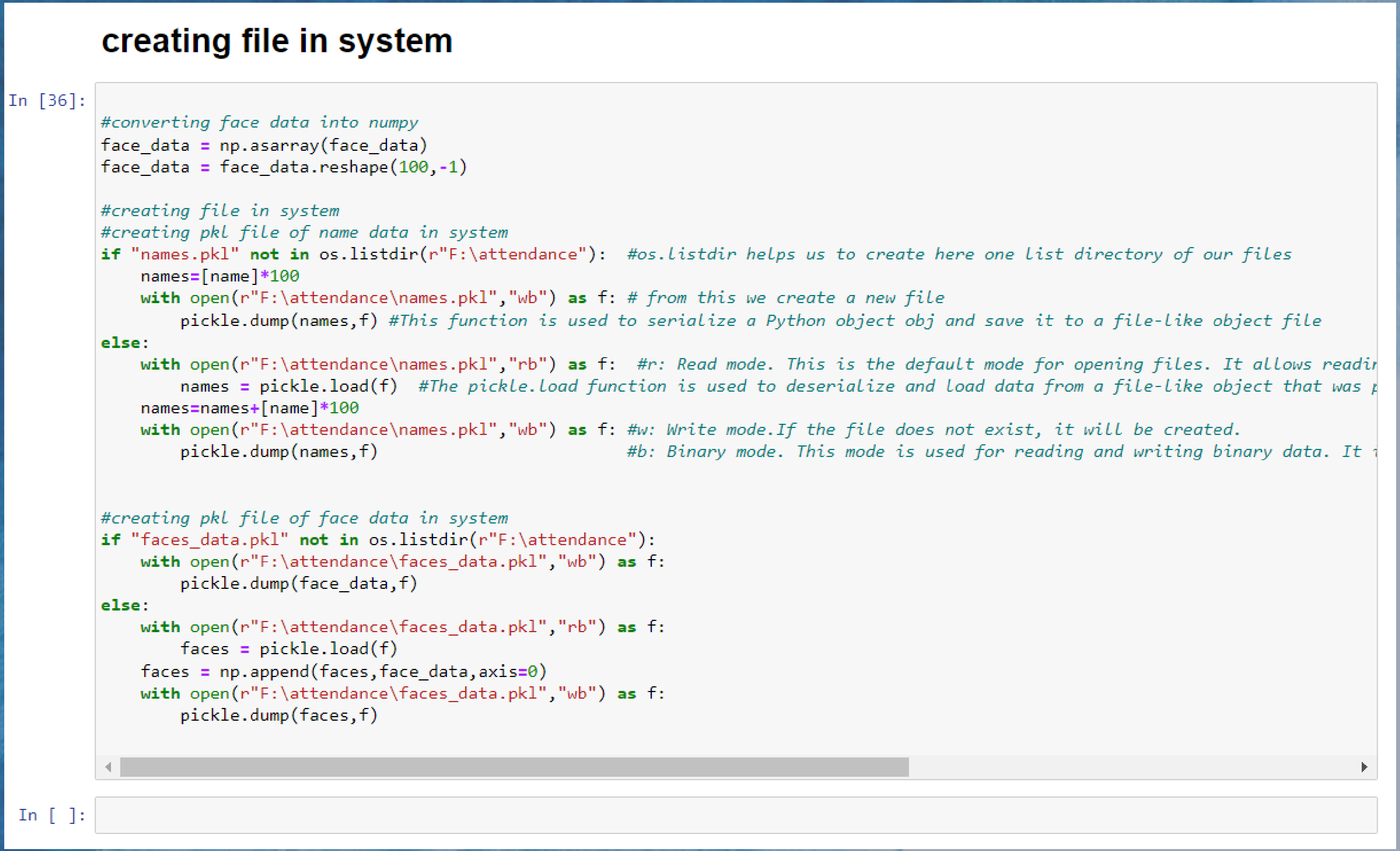
Cascade Classifier is an object detection algorithm used for identifying objects in images or video frames.

```
In [35]: face_data=[]
         i=0

         name=input("Enter Your Name:") # Username
         while True:   # for infinite loop
             ret,frame=video.read()   #video.read gives us 2 values no. 1 for Boolean below that our webcame is okay or not and 2 value i
             gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
             faces = facedetect.detectMultiScale(gray,1.3,5)      #it will give us 4dimensions (x,y)cordinates,height,width of the image,
             for (x,y,w,h) in faces:
                 crop_img = frame[y:y+h, x:x+w, :]    #this id used for crop our face from the image.

                 resized_img = cv2.resize(crop_img, (50,50))   #resize the frame

                 if len(face_data)<100 and i %10==0:
                     face_data.append(resized_img)
                 i=i+1
                 cv2.putText(frame, str(len(face_data)), (50,50), cv2.FONT_HERSHEY_DUPLEX,1,(50,50,255),1)  #The cv2.putText function is
                 cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,255),1)
             cv2.imshow("frame",frame)
             k = cv2.waitKey(1)    #It is a method to display an image in a window and wait for a keyboard event to occur before proceedi
             if k==ord("q") or len(face_data)==100: #We passing q from keyboard this infinte loop will be gone and and our video frame wi
                 break

         video.release()  # for release the video
         cv2.destroyAllWindows()



         Enter Your Name:Abhishek sain
```

# DATA COLLECTION & PREPROCESSING:

- Initialize an empty list called “face_data” to store face images.
- Prompt the user to enter their name to start data collection.
- Capture video frames and convert them to grayscale for face detection.
- Use a face detection algorithm to identify faces in the frames.
- For each detected face, crop the image and resize it to a standard size (50x50 pixels).
- Collect face data by appending the resized face image to the “face_data“list.
- Display the number of collected face images on the video frame.
- Draw rectangles around the detected faces for visualization.
- Continue the data collection until 100 face images are collected or the user presses 'q'.
- Release the video capture and close all windows

# CREATING FILE FOR AAS:



## creating file in system

In [36]:

```
#converting face data into numpy
face_data = np.asarray(face_data)
face_data = face_data.reshape(100,-1)

#creating file in system
#creating pkl file of name data in system
if "names.pkl" not in os.listdir(r"F:\attendance"):  #os.listdir helps us to create here one list directory of our files
    names=[name]*100
    with open(r"F:\attendance\names.pkl","wb") as f: # from this we create a new file
        pickle.dump(names,f) #This function is used to serialize a Python object obj and save it to a file-like object file
else:
    with open(r"F:\attendance\names.pkl","rb") as f:  #r: Read mode. This is the default mode for opening files. It allows readin
        names = pickle.load(f)  #The pickle.load function is used to deserialize and load data from a file-like object that was p
    names=names+[name]*100
    with open(r"F:\attendance\names.pkl","wb") as f: #w: Write mode.If the file does not exist, it will be created.
        pickle.dump(names,f)                         #b: Binary mode. This mode is used for reading and writing binary data. It i


#creating pkl file of face data in system
if "faces_data.pkl" not in os.listdir(r"F:\attendance"):
    with open(r"F:\attendance\faces_data.pkl","wb") as f:
        pickle.dump(face_data,f)
else:
    with open(r"F:\attendance\faces_data.pkl","rb") as f:
        faces = pickle.load(f)
    faces = np.append(faces,face_data,axis=0)
    with open(r"F:\attendance\faces_data.pkl","wb") as f:
        pickle.dump(faces,f)
```

In [ ]:

# CREATING FILE FOR AAS:

- Convert the collected “face_data” into a NumPy array and reshape it to prepare for storage.

- Make sure there's a file named "names.pkl" in the "F:\attendance" directory:
- If it's not there, create a list called "names" with the user's name repeated 100 times.
- Save this list as a file named "names.pkl" in the "F:\attendance" directory.

- Check if a file named "faces_data.pkl" exists in the "F:\attendance" directory:
  *If not, save the “face_data “as a pickle file named "faces_data.pkl" in the "F:\attendance" directory.

- If "faces_data.pkl" already exists:
  *Load the existing “faces” data from "faces_data.pkl".
  *Append the new “face_data” to the existing faces data.

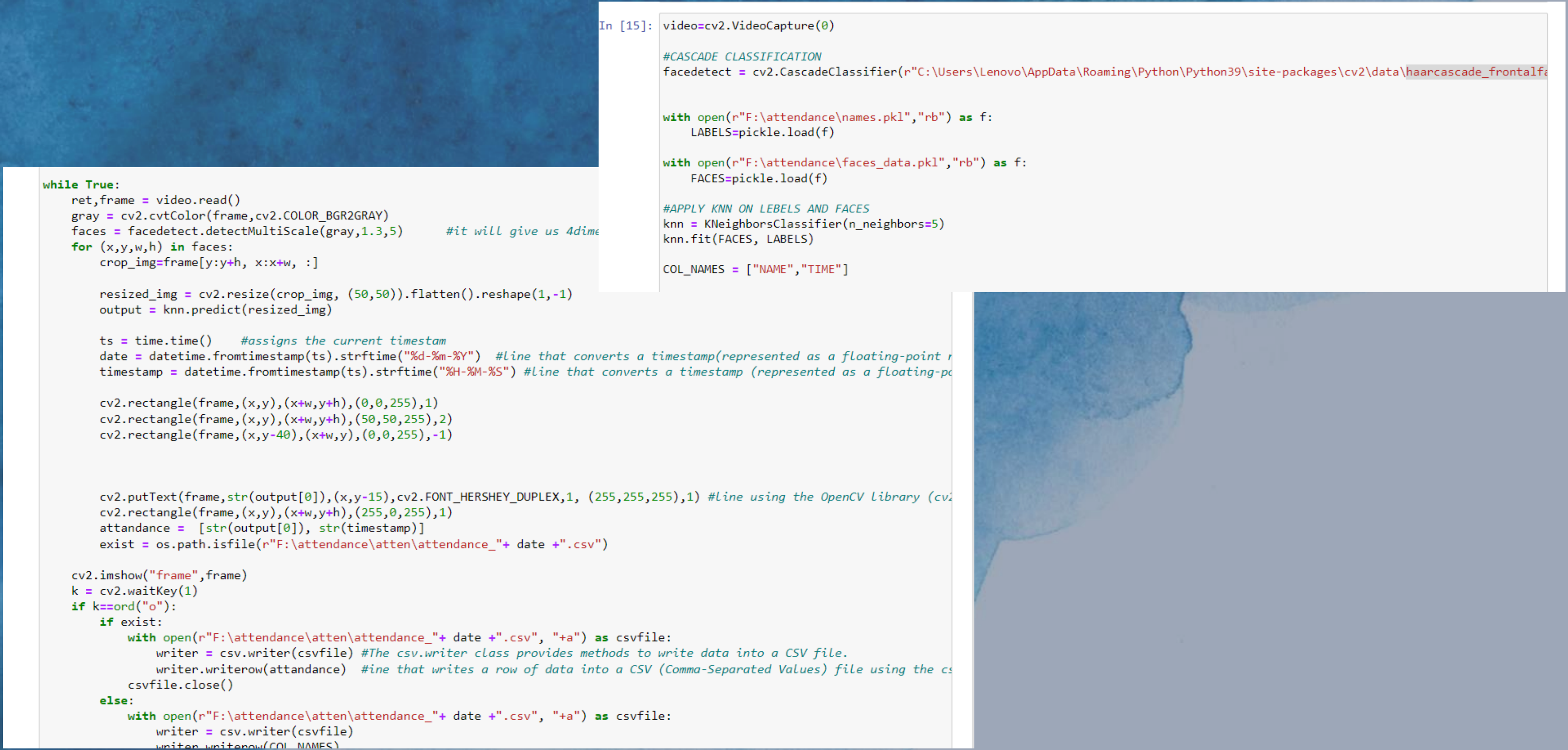
- Save the updated faces data as a pickle file named "faces_data.pkl" in the "F:\attendance" directory.

- This process ensures data persistence and allows the system to retain attendance records across sessions.

- Pickle files offer a convenient way to store and retrieve complex data structures in Python.

- The stored face data can be used for further analysis, recognition tasks, or training machine learning models.

# CLUSTERING OF DATA



```
In [15]: video=cv2.VideoCapture(0)

         #CASCADE CLASSIFICATION
         facedetect = cv2.CascadeClassifier(r"C:\Users\Lenovo\AppData\Roaming\Python\Python39\site-packages\cv2\data\haarcascade_frontalfa


         with open(r"F:\attendance\names.pkl","rb") as f:
             LABELS=pickle.load(f)

         with open(r"F:\attendance\faces_data.pkl","rb") as f:
             FACES=pickle.load(f)

         #APPLY KNN ON LEBELS AND FACES
         knn = KNeighborsClassifier(n_neighbors=5)
         knn.fit(FACES, LABELS)

         COL_NAMES = ["NAME","TIME"]
```

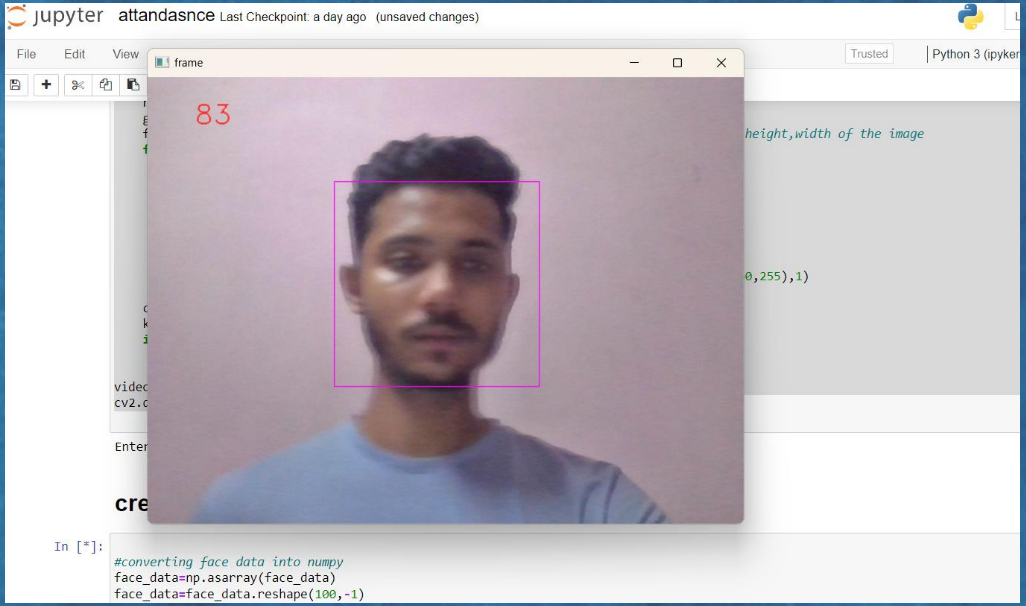
```
while True:
    ret,frame = video.read()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = facedetect.detectMultiScale(gray,1.3,5)      #it will give us 4dime
    for (x,y,w,h) in faces:
        crop_img=frame[y:y+h, x:x+w, :]

        resized_img = cv2.resize(crop_img, (50,50)).flatten().reshape(1,-1)
        output = knn.predict(resized_img)

        ts = time.time()    #assigns the current timestam
        date = datetime.fromtimestamp(ts).strftime("%d-%m-%Y")  #line that converts a timestamp(represented as a floating-point r
        timestamp = datetime.fromtimestamp(ts).strftime("%H-%M-%S") #line that converts a timestamp (represented as a floating-po

        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),1)
        cv2.rectangle(frame,(x,y),(x+w,y+h),(50,50,255),2)
        cv2.rectangle(frame,(x,y-40),(x+w,y),(0,0,255),-1)



        cv2.putText(frame,str(output[0]),(x,y-15),cv2.FONT_HERSHEY_DUPLEX,1, (255,255,255),1) #line using the OpenCV library (cv2
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,255),1)
        attandance =  [str(output[0]), str(timestamp)]
        exist = os.path.isfile(r"F:\attendance\atten\attendance_"+ date +".csv")

    cv2.imshow("frame",frame)
    k = cv2.waitKey(1)
    if k==ord("o"):
        if exist:
            with open(r"F:\attendance\atten\attendance_"+ date +".csv", "+a") as csvfile:
                writer = csv.writer(csvfile) #The csv.writer class provides methods to write data into a CSV file.
                writer.writerow(attandance)  #ine that writes a row of data into a CSV (Comma-Separated Values) file using the cs
            csvfile.close()
        else:
            with open(r"F:\attendance\atten\attendance_"+ date +".csv", "+a") as csvfile:
                writer = csv.writer(csvfile)
                writer.writerow(COL_NAMES)
```

# CLUSTERING OF DATA

- Capture video using the computer's camera to obtain real-time frames.
- Use a pre-trained Haar Cascade classifier for detecting faces in each frame.
- Load the stored user names and their corresponding face data from "names.pkl" and "faces_data.pkl".
- Apply the K-Nearest Neighbors (KNN) algorithm on the face data and corresponding labels (user names).
- Continuously analyze video frames and predict the names of detected faces using KNN.
- Capture the current timestamp and date for attendance recording.
- Display the predicted names on the frames and mark the recognized faces with a rectangle.
- Allow the user to press 'o' to record the attendance by appending it to the CSV file for the current date.
- The attendance details are stored in "attendance_date.csv" with columns "NAME" and "TIME".
- Press 'q' to exit the program and stop attendance tracking.

# DATA COLLECTION

# TAKING ATTANDANCE

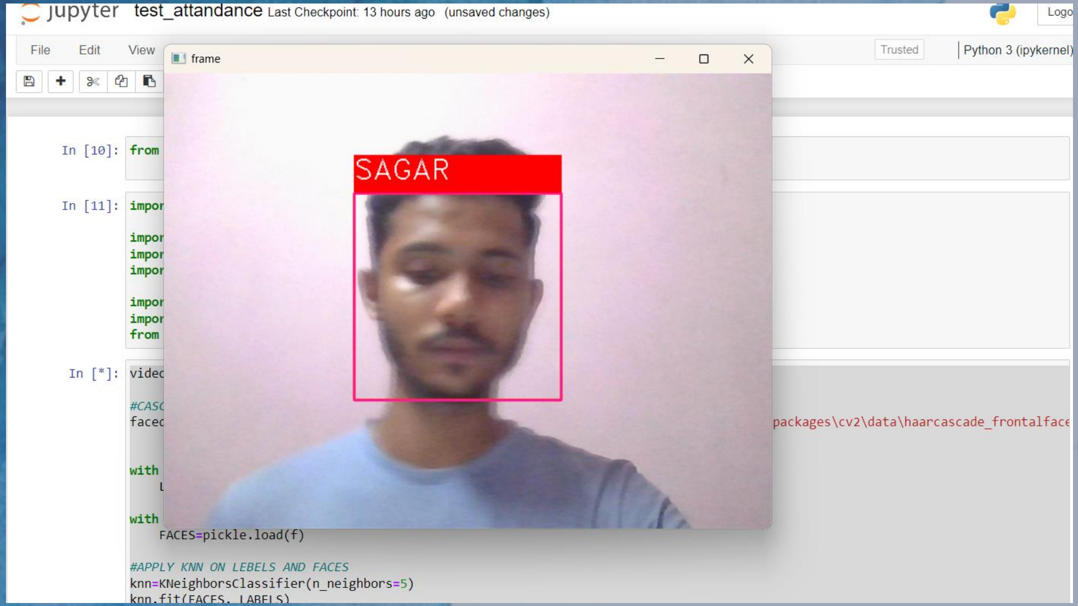


```
    height,width of the image

    0,255),1)

video
cv2.c
```

```
Enter
```

**cre**

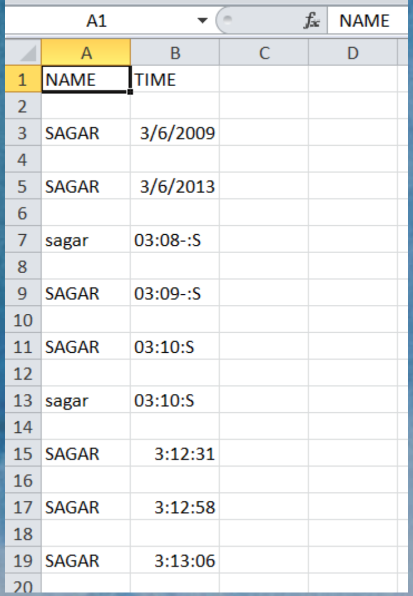
```
In [*]:
#converting face data into numpy
face_data=np.asarray(face_data)
face_data=face_data.reshape(100,-1)
```



jupyter
test_attandance Last Checkpoint: 13 hours ago (unsaved changes)
Logo
File
Edit
View
Trusted
Python 3 (ipykernel)
frame
SAGAR

```
In [10]: from

In [11]: impor

         impor
         impor
         impor

         impor
         impor
         from

In [*]:  videc

         #CASC
         facec                packages\cv2\data\haarcascade_frontalface


         with
             L

         with
             FACES=pickle.load(f)

         #APPLY KNN ON LEBELS AND FACES
         knn=KNeighborsClassifier(n_neighbors=5)
         knn.fit(FACES, LABELS)
```

# AAS RECORD



| | A | B | C | D |
|---|---|---|---|---|
| 1 | NAME | TIME | | |
| 2 | | | | |
| 3 | SAGAR | 3/6/2009 | | |
| 4 | | | | |
| 5 | SAGAR | 3/6/2013 | | |
| 6 | | | | |
| 7 | sagar | 03:08-:S | | |
| 8 | | | | |
| 9 | SAGAR | 03:09-:S | | |
| 10 | | | | |
| 11 | SAGAR | 03:10:S | | |
| 12 | | | | |
| 13 | sagar | 03:10:S | | |
| 14 | | | | |
| 15 | SAGAR | 3:12:31 | | |
| 16 | | | | |
| 17 | SAGAR | 3:12:58 | | |
| 18 | | | | |
| 19 | SAGAR | 3:13:06 | | |
| 20 | | | | |

# CONCLUSION

- The Advanced Attendance System employs computer vision techniques, including facial recognition and machine learning algorithms.
- Real-time processing capabilities enable precise and low-latency attendance tracking.
- The user interface is designed for intuitive interactions, promoting ease of use.
- The system's architecture allows seamless scalability, catering to diverse environments.
- Advanced facial recognition methods enhance security and prevent proxy attendance.
- Stored data can be utilized for in-depth analysis and recognition model training.
- The system offers potential for future enhancements and optimizations in attendance management.

# THANKYOU