MACHINE LEARNING

(Driver Drowsiness Detection)

Summer Internship Report Submitted in partial fulfillment

of the requirement for undergraduate degree of

Bachelor of Technology

In

Computer Science Engineering

By

KOKA RUCHITHA SAI RUKMINI

221710303025

Under the Guidance of

Mr.

Assistant Professor

Department of Electronics and Communication Engineering
GITAM School of Technology
GITAM (Deemed to be University)
Hyderabad-502329
July 2020

I

DECLARATION

I submit this industrial training work entitled **“Driver Drowsiness Detection**” to GITAM (Deemed to Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of “**Bachelor of Technology**” in “**Computer Science Engineering**”. I declare that it was carried out independently by me under the guidance of **Mr.** Asst. Professor, GITAM (Deemed to Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD Koka Ruchitha Sai Rukmini

Date: 221710303025

GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Dated:

CERTIFICATE

This is to certify that the Industrial Training Report entitled **"DRIVER DROWSINESS DETECTION"** is being submitted by Koka Ruchitha Sai Rukmini (221710303025) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science Engineering** at GITAM (Deemed to Be University), Hyderabad during the academic year 2020-21

It is faithful record work carried out by Koka Ruchitha Sai Rukmini at the **Computer Science Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

Mr.	**Dr.**
Assistant Professor	Professor and HOD
Department of ECE	Department of ECE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected **Dr. N. Siva Prasad,** Pro Vice Chancellor, GITAM Hyderabad and **Dr. CH. Sanjay,** Principal, GITAM Hyderabad

I would like to thank respected **Dr.** Head of the Department of Electronics and Communication Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties **Mr.** who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Koka Ruchitha Sai Rukmini

221710303025

ABSTRACT

Drowsiness is the feeling of being sleepy. It is also called as sleepiness. Sometimes the causes for sleepiness cannot be figured easily. The main causes of sleepiness are lack of sleep, working till late night, having a bad routine of sleeping at odd times, intake of drugs, medicines, alcohol, etc. The drowsiness never becomes a problem until we get to know that sometimes it may become the reason for many deaths. Now a days the main reason for accidents is driver drowsiness. Many steps were taken by the government to control road accidents like road safety measures, traffic rules, etc. but no preventive measures were taken against the drowsiness of the driver.

The project is a framework that detects the drowsiness level of driver during driving through "Machine Learning Algorithms". We use OpenCV to detect the face and eyes of a person. A CNN model is used for detecting whether the person is drowsy or not based on the amount of time he opens his eyes. The result is in the form of an accuracy score through which we can predict if the person is feeling drowsy or not. The output is in the form of a video, where the person's face and eyes are detected. The detected face and eyes are passed to the model and returns if the person is drowsy or not. If the model returns that the person is drowsy then an alarm is popped to alert the person that he is sleepy.

Table of Contents:

TABLE OF FIGURES

CHAPTER 1

MACHINE LEARNING

1.1 INTRODUCTION:

Machine Learning (ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence (AI).

1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and "more items to consider" and "get yourself a little something" on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today's data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that's in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works

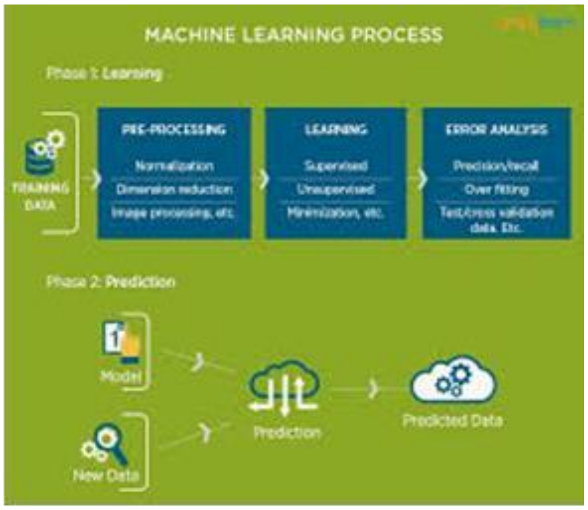


Figure 1.2: The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let’s consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

1.4.1 Supervised Learning:

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

1.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

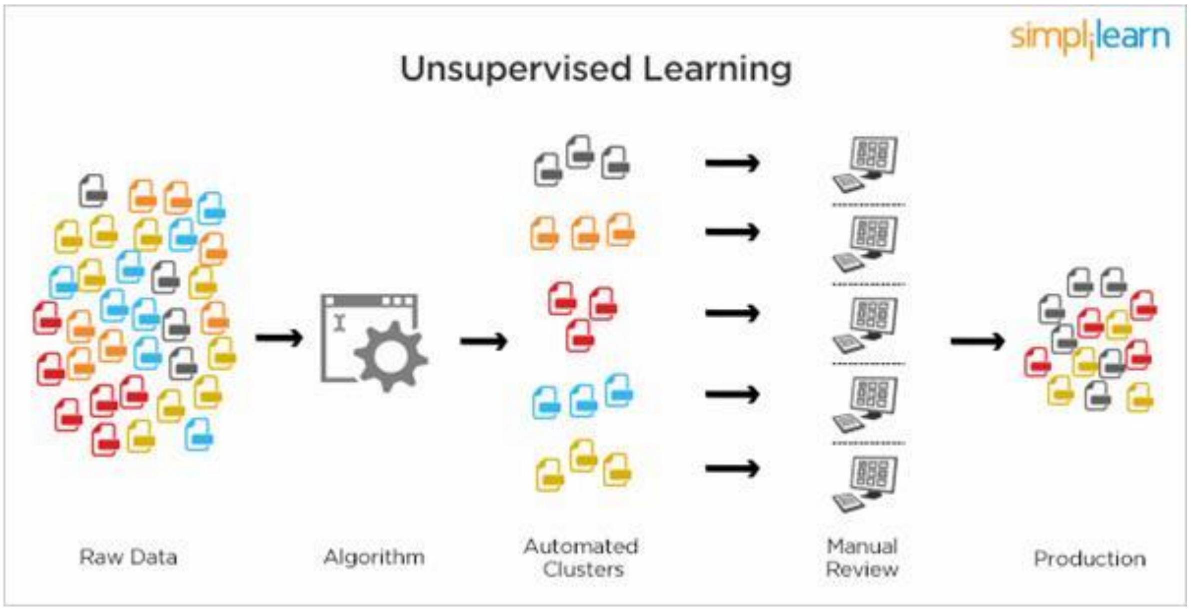


Figure 1.4.2: Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

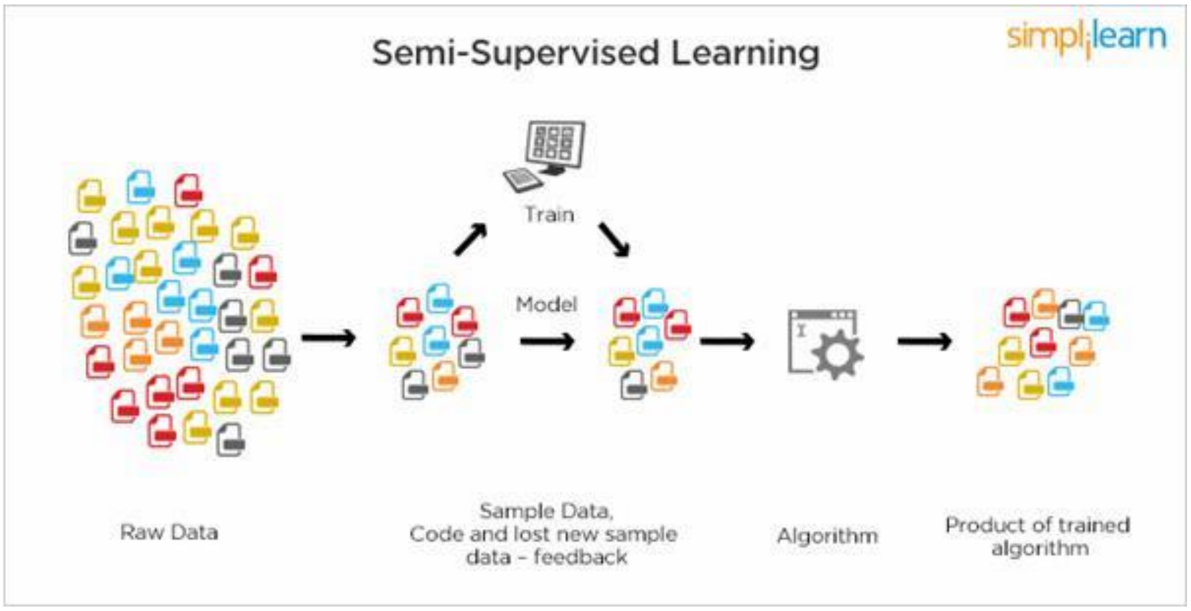

Semi-Supervised Learning
simplilearn
Train
Model
Raw Data
Sample Data, Code and lost new sample data - feedback
Algorithm
Product of trained algorithm

Figure 1.4.3: Semi Supervised Learning

1.5 RELATION BETWEEN DATA MINING, MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER 2

PYTHON

Basic programming language used for machine learning is: PYTHON

2.1 INTRODUCTION TO PYHTON:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general-purpose programming language that is often applied in scripting roles
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

2.2 HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990's
- Its latest version is 3.7, it is generally called as python3

2.3 FEATURES OF PYTHON:

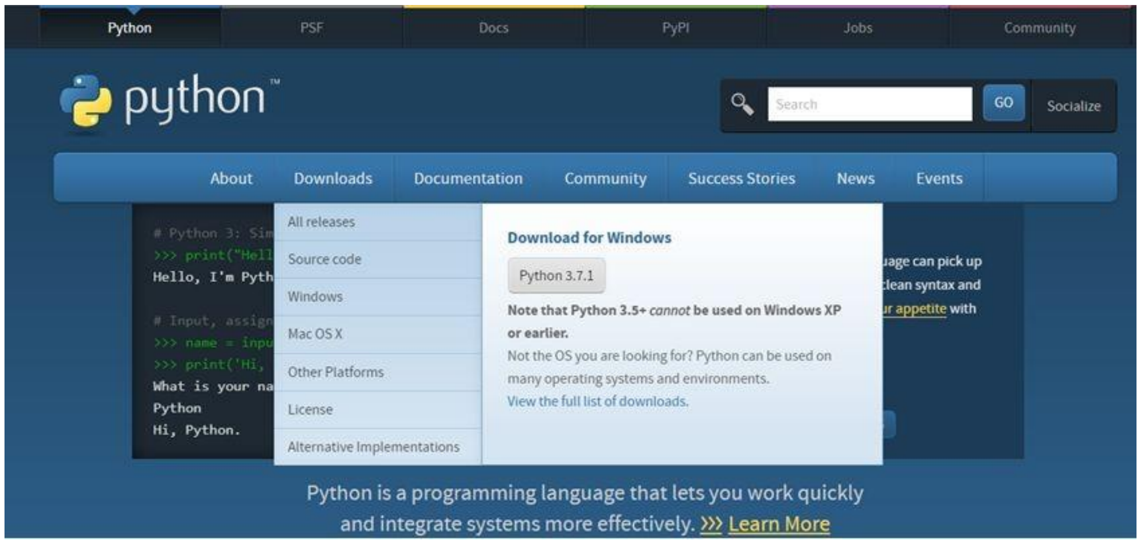
- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.
- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

2.3 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.
- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

2.4.1 Installation (using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.



```
# Python 3: Sim
>>> print("Hell
Hello, I'm Pyth

# Input, assign
>>> name = inpu
>>> print('Hi,
What is your na
Python
Hi, Python.
```

Download for Windows

Note that Python 3.5+ *cannot* be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

uage can pick up
lean syntax and
ur appetite with

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Figure 2.4.1: Python download

2.4.2 Installation (using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager quickly installs and manages packages

- In WINDOWS:
- In windows
  - Step 1: Open Anaconda.com/downloads in web browser.
  - Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
  - Step 3: select installation type (all users)
  - Step 4: Select path (i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
  - Step 5: Open jupyter notebook (it opens in default browser)

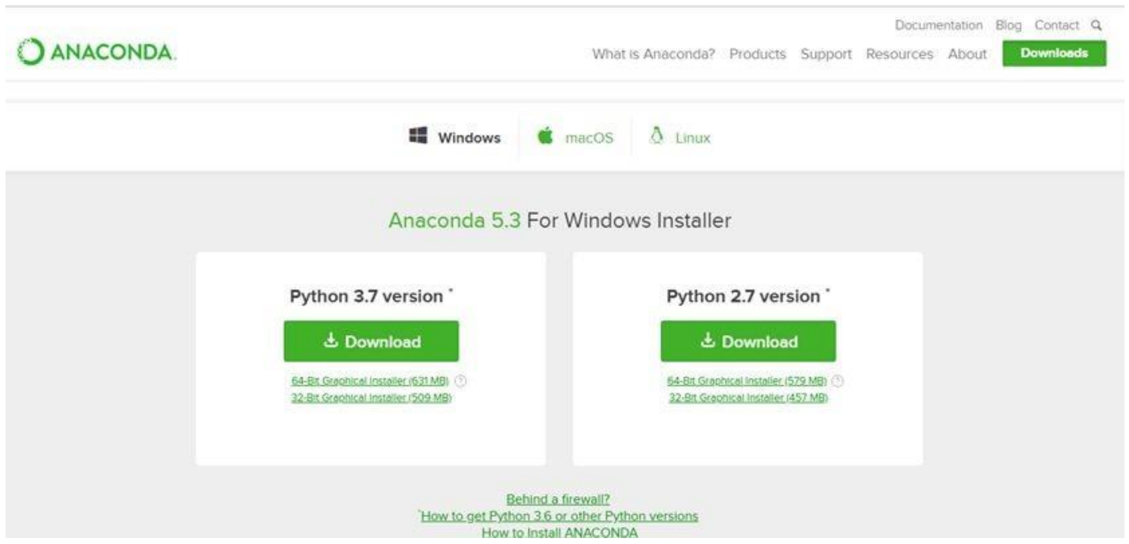

ANACONDA
Documentation Blog Contact
What is Anaconda? Products Support Resources About Downloads
Windows macOS Linux
Anaconda 5.3 For Windows Installer
Python 3.7 version
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (509 MB)
Python 2.7 version
Download
64-Bit Graphical Installer (579 MB)
32-Bit Graphical Installer (457 MB)
Behind a firewall?
How to get Python 3.6 or other Python versions
How to Install ANACONDA

Figure 2.4.2: Anaconda download

Figure 2.4.3: Jupyter notebook

2.4 PYTHON VARIABLE TYPES:

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

- Python has five standard data types –
    - Numbers
    - Strings
    - Lists
    - Tuples
    - Dictionary

2.5.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

2.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets ([]).
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

2.5.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated.
- Tuples can be thought of as read-only lists.
- For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

2.5.5 Python Dictionary:

- Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({}) and values can be assigned and accessed using square braces ([]).
- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

2.5 PYTHON FUNCTION:

2.6.1 Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e. ()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

2.6.2 Calling a Function:

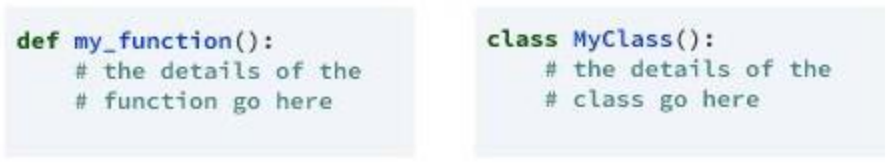
Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

2.6 PYTHON USING OOP’s CONCEPTS:

2.7.1 Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- Data member: A class variable or instance variable that holds data associated with a class and its objects.
- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.
- Defining a Class:
    - We define a class in a very similar way how we define a function.
    - Just like a function, we use parentheses and a colon after the class name (i.e. () :) when we define a class. Similarly, the body of our class is

indented like a functions body is.



```
def my_function():
    # the details of the
    # function go here
```

```
class MyClass():
    # the details of the
    # class go here
```

Figure 2.7.1: Defining a Class

2.7.2 __init__method in Class:

- The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.
- The init method has a special name that starts and ends with two underscores: init ().

CHAPTER 3

CASE STUDY

3.1 PROBLEM STATEMENT:

To predict driver drowsiness using OpenCV. We detect the drowsiness and develop an alerting system for sleepiness of driver in order to reduce the risk of accidents.

3.2 DATA SET:

We have used the dlib's shape predictor and analyzed the input images with a pre-trained file, '_shape_predictor_68_face_landmarks.dat' to get the facial landmarks detected. The trained file is an implementation of the _One Millisecond Face Alignment with an Ensemble of Regression Trees '. To train the detector they used a manually labelled facial landmarks specifying specific (x, y)- coordinates of regions surrounding each of the facial structure. An ensemble of regression trees is used to train the file for estimation of the facial landmark positions through pixel intensities themselves. The facial landmark detector is used to estimate the location of 68 (x, y)-coordinates that map to facial structure on the face.

3.3 OBJECTIVE OF THE CASE STUDY:

The objective of the project is to develop a drowsiness detection system along with the alert system which detects the sleepiness of the driver by extracting the features from the face and process the data and if drowsiness is detected a warning is given to the driver in the form of an alarm.

3.4 TECHNOLOGY:

TensorFlow:

It is a free platform for ML. It is an end to end platform. It has a pliable atmosphere of tools, comprehensive resources that let the programmers to build applications easily and quickly. It is a symbolic math library, used for machine learning applications such as neural networks.

OpenCV:

It is a library of python binding designed to solve all the vision related problems of the computer like detecting the face, eyes etc. Python is a programming language which allows the user who is using to write very few lines of code for large programs. OpenCV is a library which contains many features of python like numpy, etc which can be used for further alcutions or operations. NumPy is a library which contains all the features related to mathematical operations like arrays, multi-dimensional array and all of the operations that are supported by NumPy has a similar syntax to that of MATLAB so that we can practice it using programming also. All the arrays we use in the OpenCV are converted as NumPy arrays or they can be imported from NumPy arrays by importing the NumPy package.

3.5 IMPLEMENTATION:

To implement the model, we used five steps namely image acquiring, face recognition, eye detection, eye tracking, and drowsy recognition.

IMAGE ACQUIRING:

The first step in the functionality of the proposed system is image acquisions i.e., recording a video by using a webcam. Then converting the video into an order of images or frames for further recognition.

FACE RECOGNITION:

It is process of recognizing faces of visual media i.e., images and the video. The face that is detected is recognized at particular position with integrated size. The landmarks for detected face are known.

EYES DETECTION:

- As eyes provide different face related features, eye recognition plays an important role in the recognition of face.
- However, some effects like dark shadows and changes in light make it difficult task to detect eyes properly in facial images.
- Eye detection has a crucial role in recognizing the sleepiness of a person.
- Once the face is detected, landmarks for eyes are detected using some specific functions, and then the distance between eyes is calculated so as to classify whether eyes are open or close.
- EAR is the most important variable in defining the position of eye. It is a variable to determine eye opened stage
- It reduces to zero when eyes are closed and increases rapidly when eyes are opened

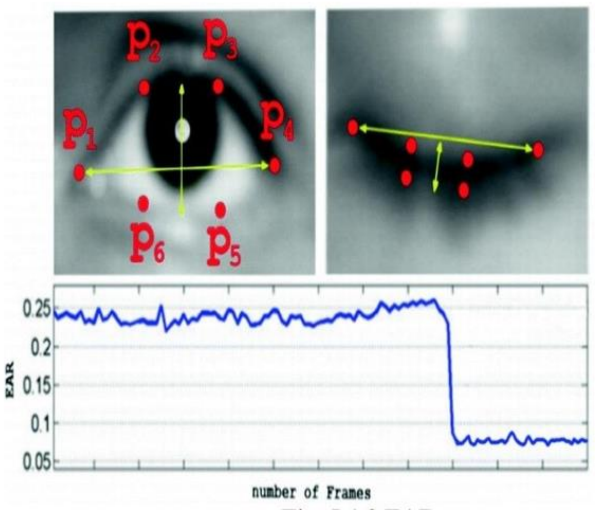


Fig 3.5: Eye Aspect Ratio

- EAR can be found by the formula:

$$\text{EAR} = \frac{\| D2 - D6 \| + \| D3 - D5 \|}{2 \| D1 - D4 \|}$$

Where $\| D2 - D6 \|$ and $\| D3 - D5 \|$ are vertical distances and $\| D1 - D4 \|$ is horizontal distance.

EYE TRACKING:

- It is a process of tracking the movements of eye and the position of eye.
- The devices used for eye tracking is called eye trackers.
- In this project we record eye moments through a video using a webcam.

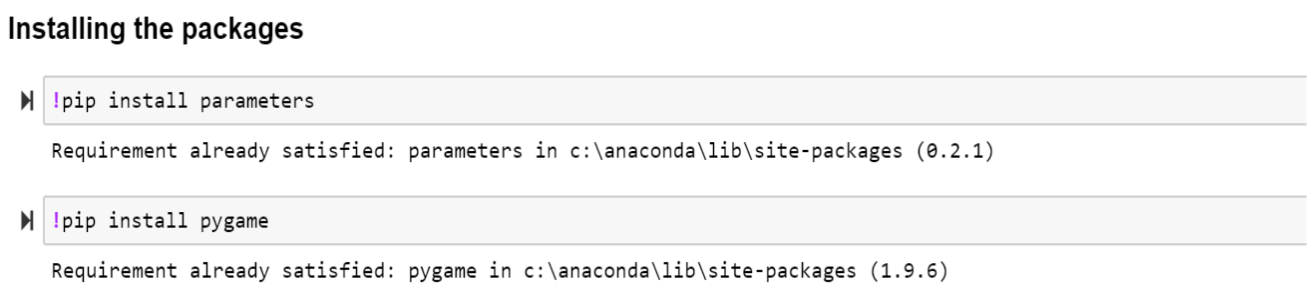
DROWSINESS DETECTION:

- This is the final step in the architecture of drowsiness detection.
- After performing all the above steps including the yawning detection i.e., movement of the mouth, drowsiness is detected based on the threshold values.
- The drowsiness is detected using the OpenCV.

CHAPTER 4

MODEL BUILDING

4.1 INSTALLING THE PACKAGES:



Installing the packages

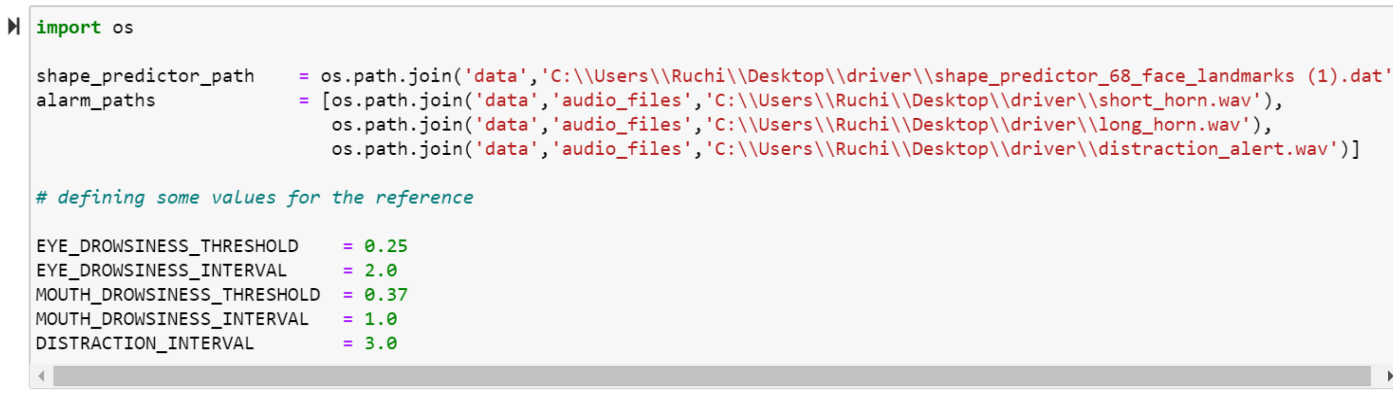
```
!pip install parameters
Requirement already satisfied: parameters in c:\anaconda\lib\site-packages (0.2.1)

!pip install pygame
Requirement already satisfied: pygame in c:\anaconda\lib\site-packages (1.9.6)
```

Figure 4.1: Installing packages

We have installed the required packages parameters Nd pygame.

- **Pygame:** Pygame is a Python wrapper module for the SDL multimedia library. It contains python functions and classes that will allow you to use SDL's support for playing cdroms, audio and video output, and keyboard, mouse and joystick input.

4.2 PARAMETERS:



```
import os

shape_predictor_path    = os.path.join('data','C:\\Users\\Ruchi\\Desktop\\driver\\shape_predictor_68_face_landmarks (1).dat'
alarm_paths             = [os.path.join('data','audio_files','C:\\Users\\Ruchi\\Desktop\\driver\\short_horn.wav'),
                           os.path.join('data','audio_files','C:\\Users\\Ruchi\\Desktop\\driver\\long_horn.wav'),
                           os.path.join('data','audio_files','C:\\Users\\Ruchi\\Desktop\\driver\\distraction_alert.wav')]

# defining some values for the reference

EYE_DROWSINESS_THRESHOLD    = 0.25
EYE_DROWSINESS_INTERVAL     = 2.0
MOUTH_DROWSINESS_THRESHOLD  = 0.37
MOUTH_DROWSINESS_INTERVAL   = 1.0
DISTRACTION_INTERVAL        = 3.0
```

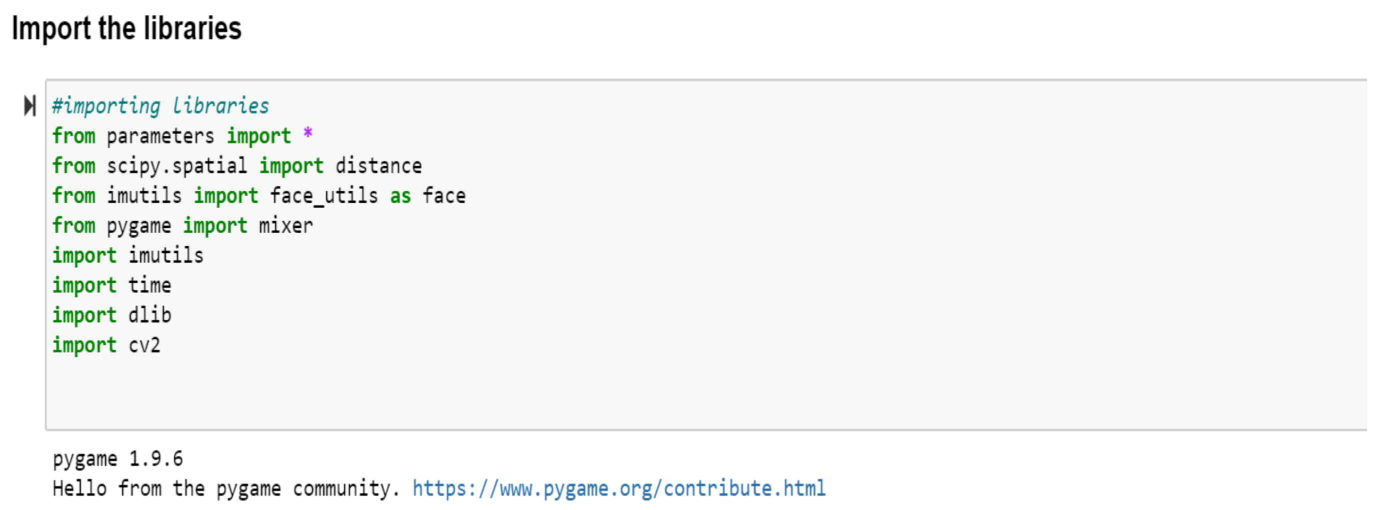
Figure 4.2: Parameters

- In this module all the required parameters are passed like eye drowsiness threshold, eye drowsiness interval, mouth drowsiness threshold, mouth drowsiness interval, and distraction interval.
- Each parameter has a specific value and for specific functionality.
- Eye drowsiness parameters are used for eye blinking detection and mouth drowsiness values are used for yawning detection and distraction interval specifies the amount of time.
- Also, we have passed three alarms, shorthorn, longhorn, distraction alert.

Dataset Link: https://drive.google.com/file/d/18o_9SXqe-XgcYFwlkse5CIe132G5TyxH/view?usp=sharing

4.3 IMPORITNG THE LIBRARIES:

We have to import libraries based on the project requirement:



Import the libraries

```
#importing libraries
from parameters import *
from scipy.spatial import distance
from imutils import face_utils as face
from pygame import mixer
import imutils
import time
import dlib
import cv2
```

```
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
```

Figure 4.3: Importing libraries

- In this module we import all required libraries such as dlib,imutils,cv2 and time.
- The parameters passed in the module are imported here.

Packages used:

Parameters: version '0.2.1'

SciPy: version '1.4.1'

Imutils: version '0.5.3'

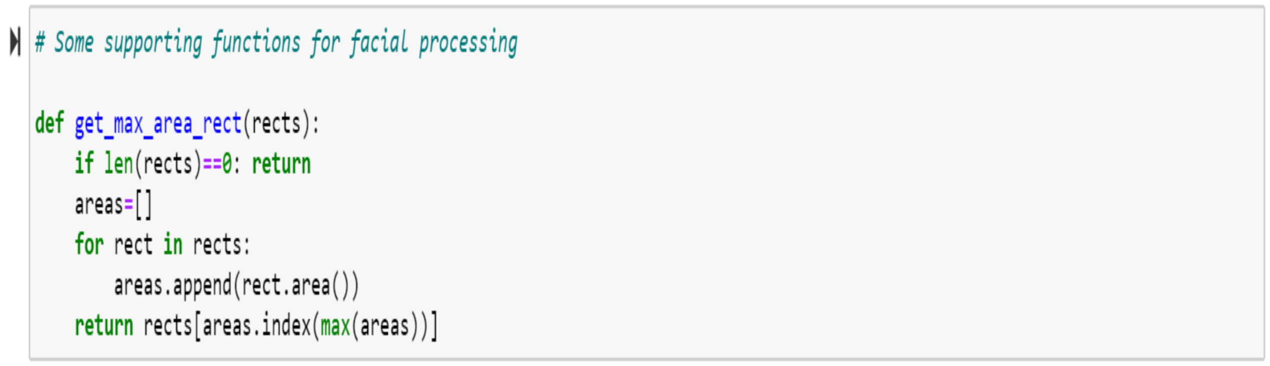
Pygame: version '1.9.6'

Dlib: version '19.20.0'

Cv2: version '4.3.0'

4.4 FACE DETECTION:

Supporting functions for face detection:



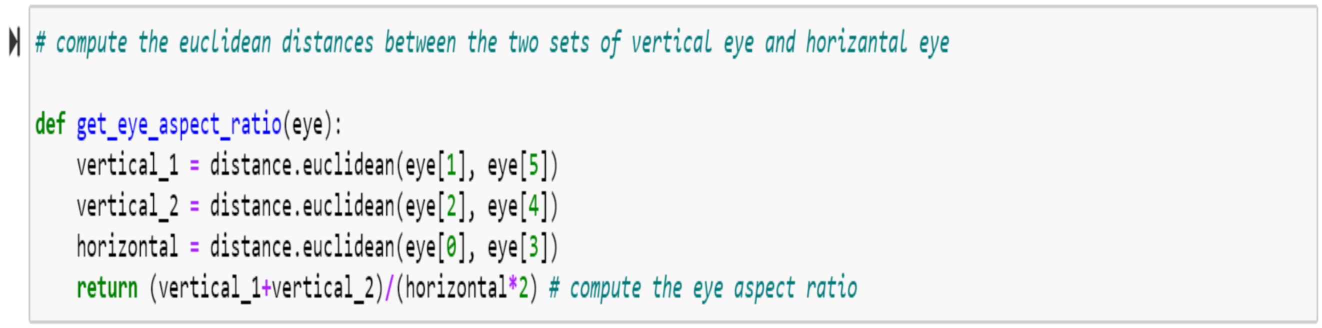
```
# Some supporting functions for facial processing

def get_max_area_rect(rects):
    if len(rects)==0: return
    areas=[]
    for rect in rects:
        areas.append(rect.area())
    return rects[areas.index(max(areas))]
```

Figure 4.4: Face Detection

- The first and the foremost method in our system is to recognize a face. So as to recognize a face we defined a function named get_max_area_rect which returns the detected face visual.

4.5 EYE ASPECT RATIO:

We also need to define the get_eye_aspect_ratio function which is used to compute the ratio of distances between the vertical eye landmarks and the distances between the horizontal eye landmarks:



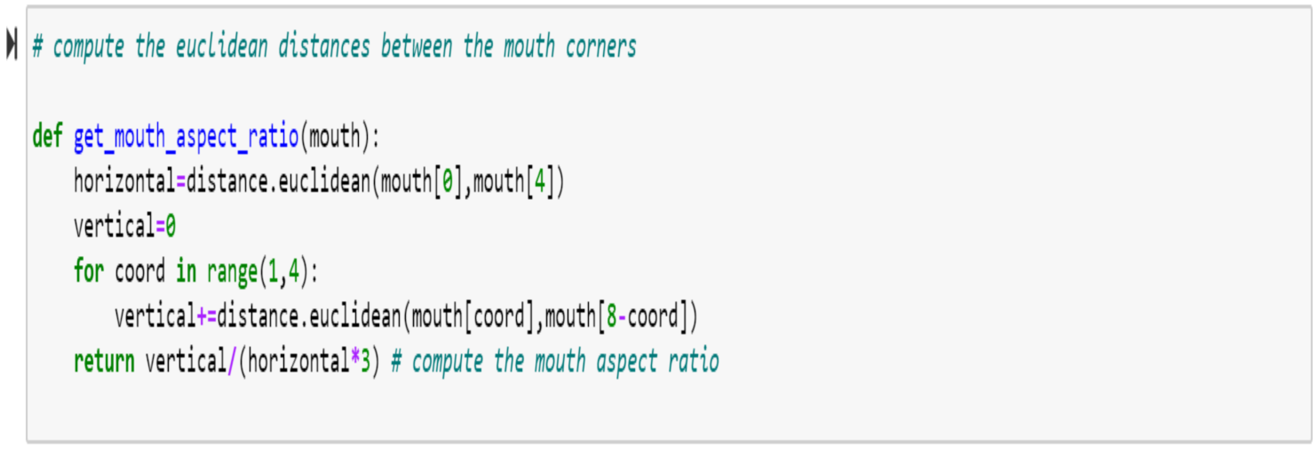
```
# compute the euclidean distances between the two sets of vertical eye and horizantal eye

def get_eye_aspect_ratio(eye):
    vertical_1 = distance.euclidean(eye[1], eye[5])
    vertical_2 = distance.euclidean(eye[2], eye[4])
    horizontal = distance.euclidean(eye[0], eye[3])
    return (vertical_1+vertical_2)/(horizontal*2) # compute the eye aspect ratio
```

Figure 4.5: Eye aspect ratio

- EAR that plays an important role in monitoring the movement of eyes.
- **Aspect ratio**: Aspect ratio is an image projection attribute that describes the proportional relationship between the width and height of an image, in this case eye.
- The aspect ratio is generally constant when the eye is open and starts tending to zero while closing of eye. Since eye blinking is performed by both eyes synchronously the aspect ratio of both eyes is averaged.
- The return value of the eye aspect ratio will be approximately constant when the eye is open. The value will then rapidly decrease towards zero during a blink.
- If the eye is closed, the eye aspect ratio will again remain approximately constant, but will be much smaller than the ratio when the eye is open.

4.6 MOUTH ASPECT RATIO:

We also need to define the get_mouth_aspect_ratio function which is used to compute the ratio of distances
between the mouth:



```
# compute the euclidean distances between the mouth corners

def get_mouth_aspect_ratio(mouth):
    horizontal=distance.euclidean(mouth[0],mouth[4])
    vertical=0
    for coord in range(1,4):
        vertical+=distance.euclidean(mouth[coord],mouth[8-coord])
    return vertical/(horizontal*3) # compute the mouth aspect ratio
```

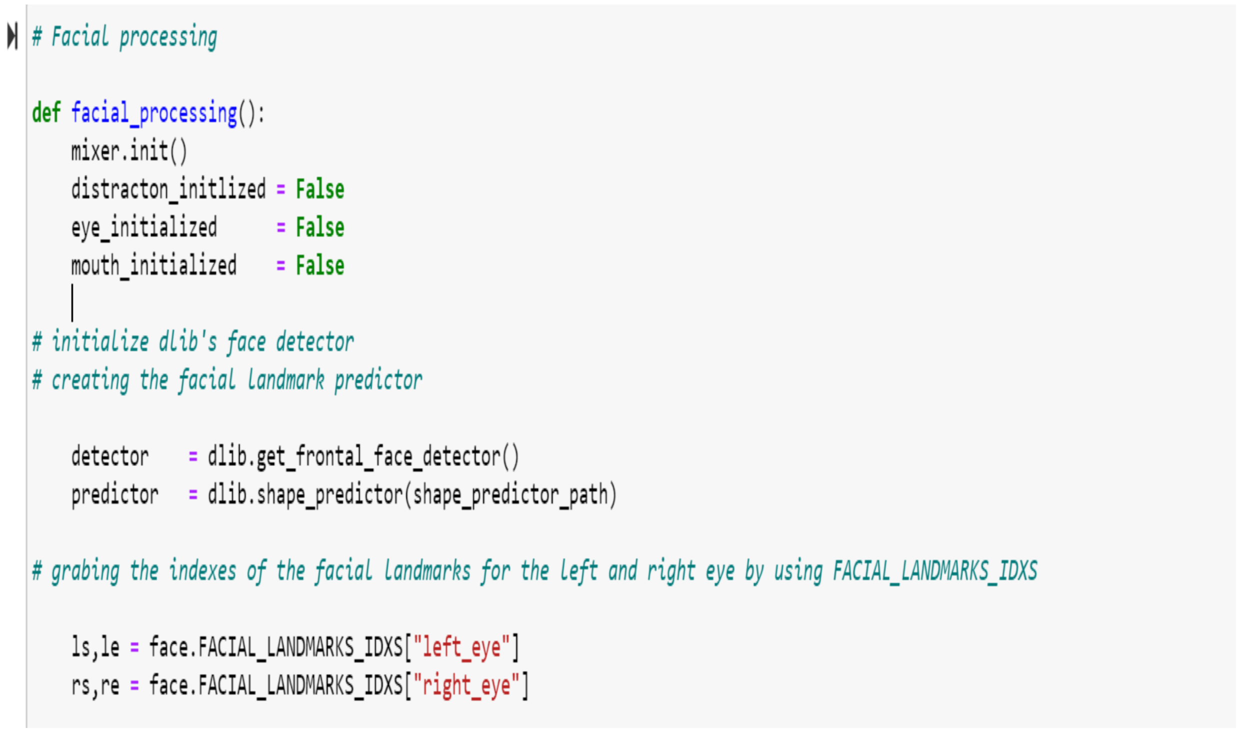
Figure 4.6: Mouth aspect ratio

- MAR plays an important role in monitoring the movement of mouth.

- So, to calculate the mouth aspect ratio we define a function called get_mouth_aspect_ratio which returns the mouth aspect ratio which helps in yawning detection

4.7 FACE PROCESSING:

4.7.1 FACIAL PROCESSING:

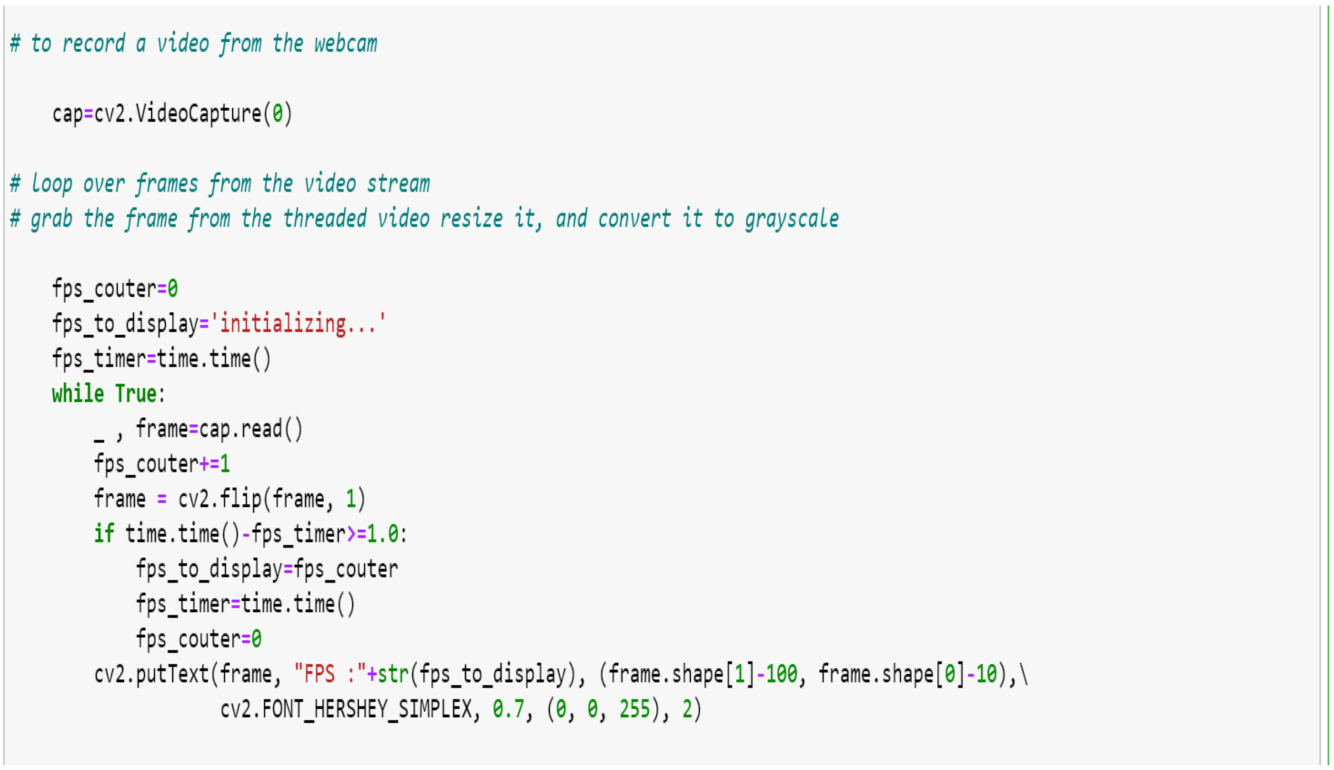


```
# Facial processing

def facial_processing():
    mixer.init()
    distracton_initlized = False
    eye_initialized      = False
    mouth_initialized    = False
    
# initialize dlib's face detector
# creating the facial landmark predictor

    detector    = dlib.get_frontal_face_detector()
    predictor   = dlib.shape_predictor(shape_predictor_path)

# grabing the indexes of the facial landmarks for the left and right eye by using FACIAL_LANDMARKS_IDXS

    ls,le = face.FACIAL_LANDMARKS_IDXS["left_eye"]
    rs,re = face.FACIAL_LANDMARKS_IDXS["right_eye"]
```

Figure 4.7.1: Face processing

- We initialized dlib’s face detector using get_frontal_face_detector and created the facial landmark predictor using the shape predictor.
- We used mixer. init () method for the alarm sound.
- Now to extract the eye regions from a set of facial landmarks, we used array slice indexes.
- Using these indexes, we’ll easily be able to extract the eye regions via an array slice.

4.7.2 VIDEO CAPTURING:

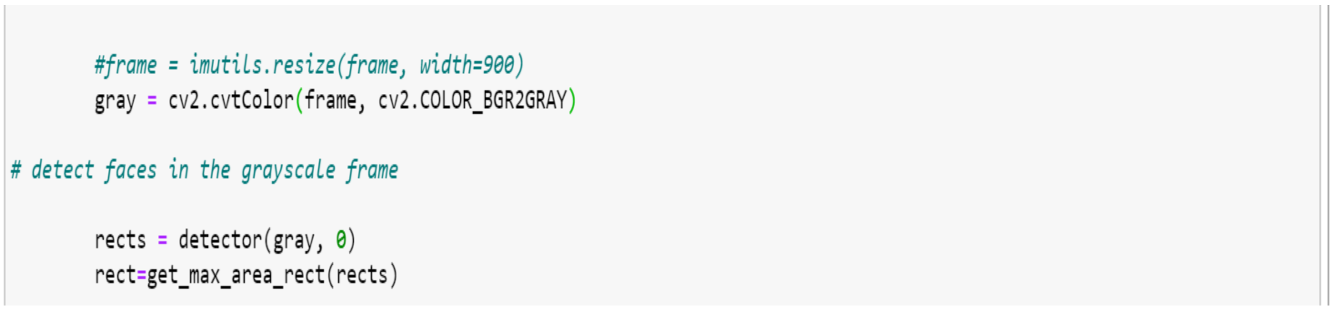


```
# to record a video from the webcam

    cap=cv2.VideoCapture(0)

# loop over frames from the video stream
# grab the frame from the threaded video resize it, and convert it to grayscale

    fps_couter=0
    fps_to_display='initializing...'
    fps_timer=time.time()
    while True:
        _ , frame=cap.read()
        fps_couter+=1
        frame = cv2.flip(frame, 1)
        if time.time()-fps_timer>=1.0:
            fps_to_display=fps_couter
            fps_timer=time.time()
            fps_couter=0
        cv2.putText(frame, "FPS :"+str(fps_to_display), (frame.shape[1]-100, frame.shape[0]-10),\
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

Figure 4.7.2: Video Capturing

- We use the cv2.VideoCapture(0) to get a video capture object for the camera.
- Set up an infinite while loop and use the read () method to read the frames using the above created object.

4.7.3 FRAME RESIZING:

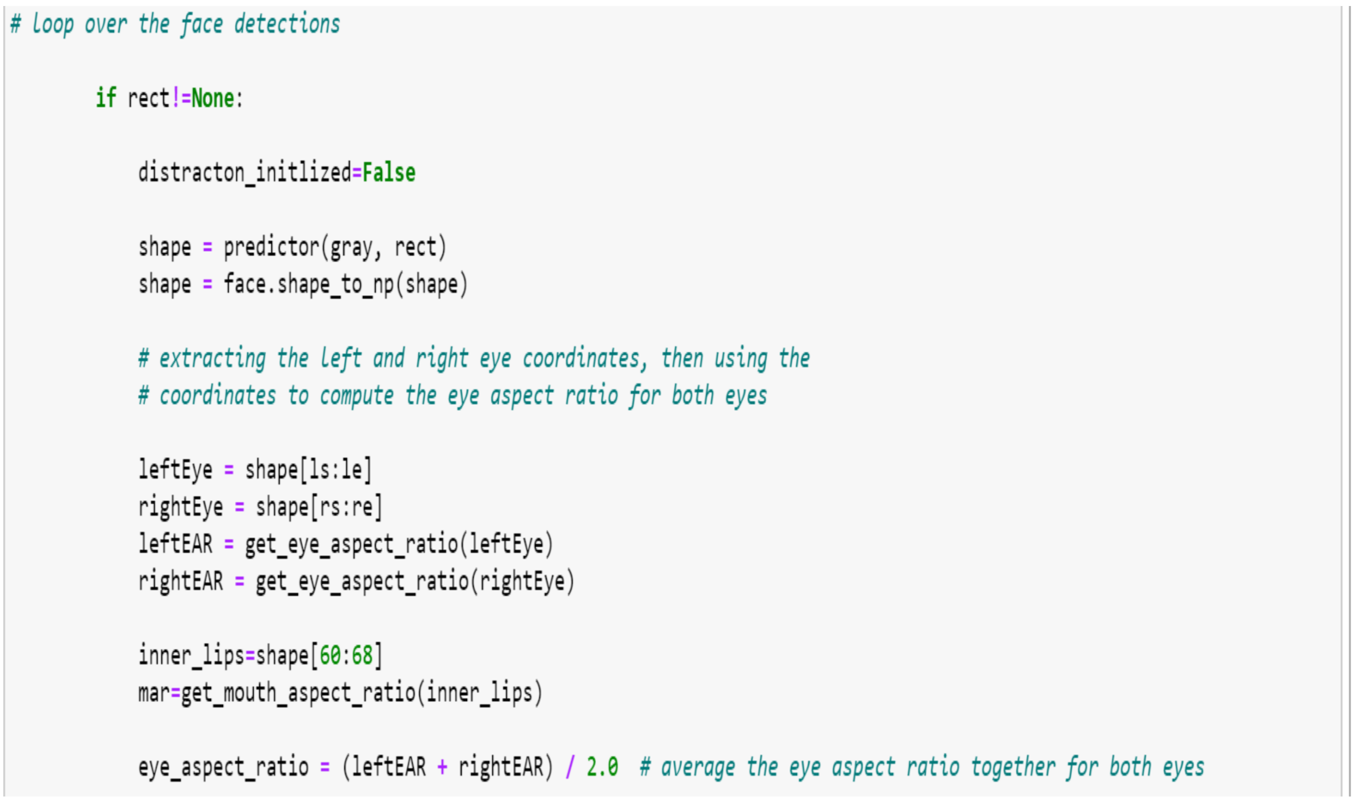


```
        #frame = imutils.resize(frame, width=900)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

# detect faces in the grayscale frame

        rects = detector(gray, 0)
        rect=get_max_area_rect(rects)
```

Figure 4.7.3 Frame Resizing

- We used the cv2.cvtColor to convert the scale to grayscale image.

4.7.4 FACE DETECTION:



```
# loop over the face detections

        if rect!=None:

            distracton_initlized=False

            shape = predictor(gray, rect)
            shape = face.shape_to_np(shape)

            # extracting the left and right eye coordinates, then using the
            # coordinates to compute the eye aspect ratio for both eyes

            leftEye = shape[ls:le]
            rightEye = shape[rs:re]
            leftEAR = get_eye_aspect_ratio(leftEye)
            rightEAR = get_eye_aspect_ratio(rightEye)

            inner_lips=shape[60:68]
            mar=get_mouth_aspect_ratio(inner_lips)

            eye_aspect_ratio = (leftEAR + rightEAR) / 2.0  # average the eye aspect ratio together for both eyes
```

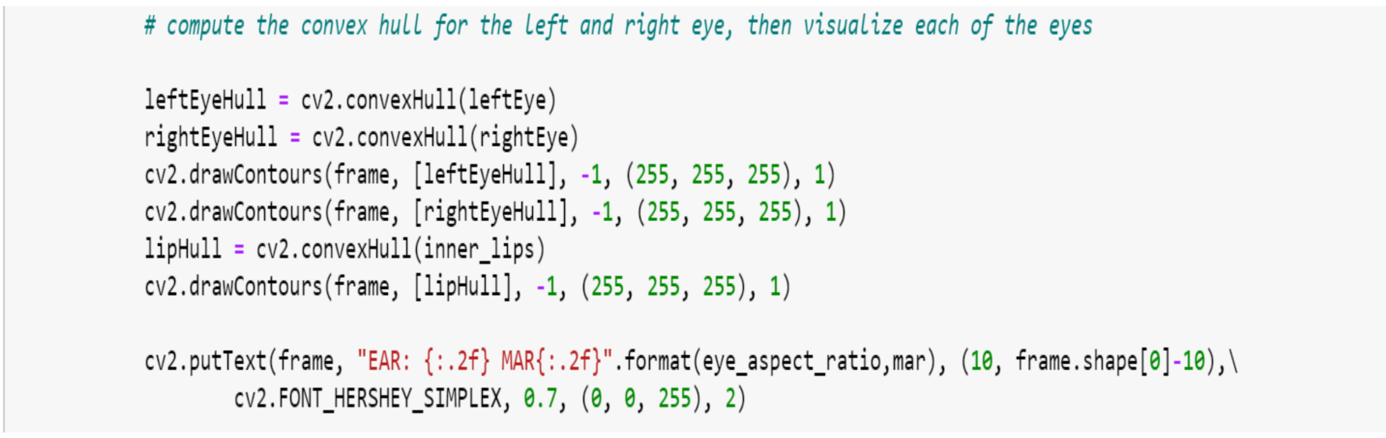
Figure 4.7.4: Face Detection

- For each of the detected faces, we apply dlib's facial landmark detector and convert the result to a NumPy array.

- Using NumPy array slicing we can extract the *(x, y)*-coordinates of the left and right eye, respectively.

- We can then *visualize* each of the eye regions on our frame by using cv2.drawContours function.

4.7.5 COMPUTE THE CONVEX HULL:



```
# compute the convex hull for the left and right eye, then visualize each of the eyes

leftEyeHull = cv2.convexHull(leftEye)
rightEyeHull = cv2.convexHull(rightEye)
cv2.drawContours(frame, [leftEyeHull], -1, (255, 255, 255), 1)
cv2.drawContours(frame, [rightEyeHull], -1, (255, 255, 255), 1)
lipHull = cv2.convexHull(inner_lips)
cv2.drawContours(frame, [lipHull], -1, (255, 255, 255), 1)

cv2.putText(frame, "EAR: {:.2f} MAR{:.2f}".format(eye_aspect_ratio,mar), (10, frame.shape[0]-10),\
        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

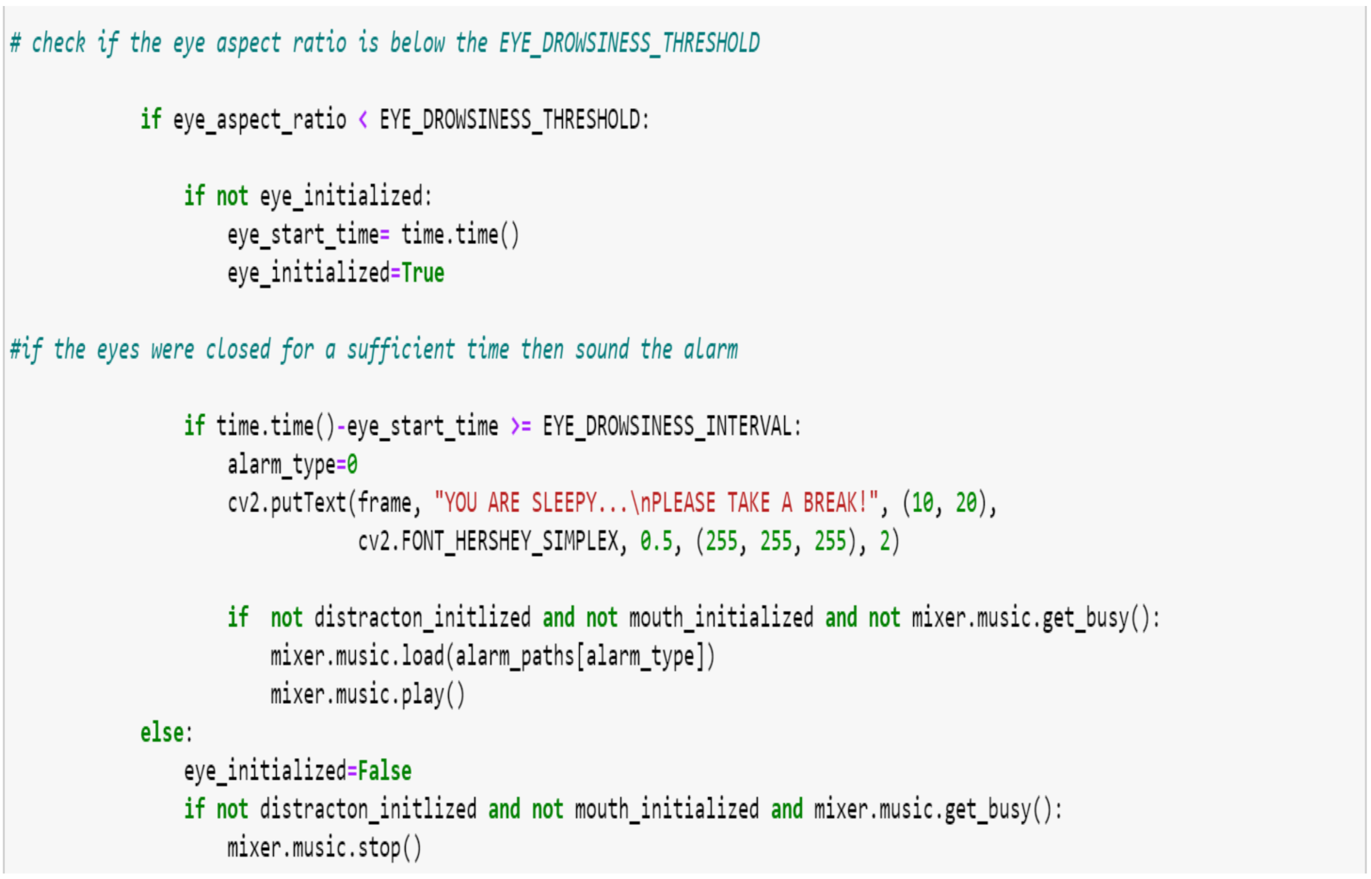
Figure 4.7.5: Convex hull

- compute the convex hull for the left and right eye, then and visualize each of the eye.

CHAPTER 5

5.1 ALERTING:

5.1.1 ALARM 1:

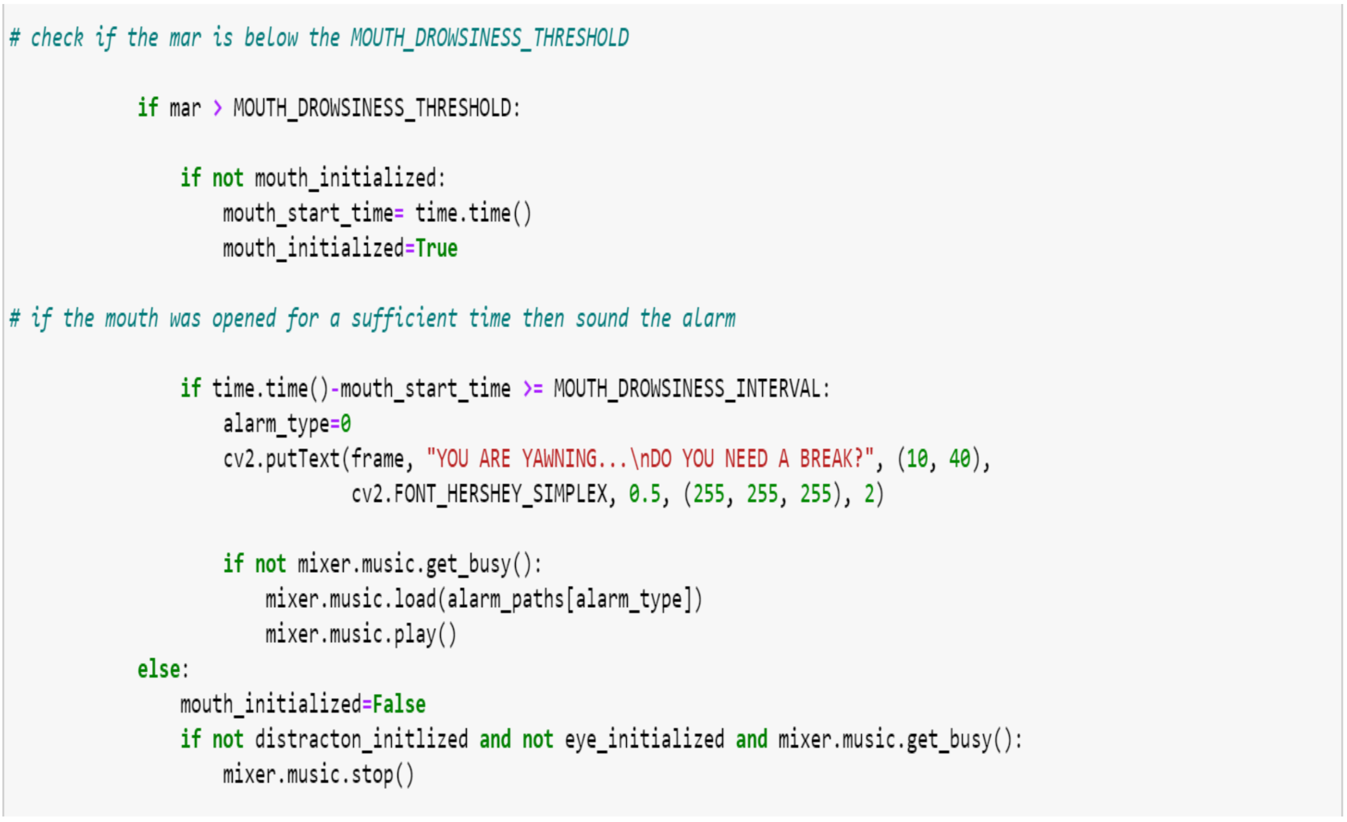


```
# check if the eye aspect ratio is below the EYE_DROWSINESS_THRESHOLD

            if eye_aspect_ratio < EYE_DROWSINESS_THRESHOLD:

                if not eye_initialized:
                    eye_start_time= time.time()
                    eye_initialized=True

#if the eyes were closed for a sufficient time then sound the alarm

                if time.time()-eye_start_time >= EYE_DROWSINESS_INTERVAL:
                    alarm_type=0
                    cv2.putText(frame, "YOU ARE SLEEPY...\nPLEASE TAKE A BREAK!", (10, 20),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

                    if  not distracton_initlized and not mouth_initialized and not mixer.music.get_busy():
                        mixer.music.load(alarm_paths[alarm_type])
                        mixer.music.play()
            else:
                eye_initialized=False
                if not distracton_initlized and not mouth_initialized and mixer.music.get_busy():
                    mixer.music.stop()
```

Figure 5.1.1 alarm1

- Our first alarm will be popped when the calculated eye aspect ratio is less than the eye_threshold_ratio.
- The alarm tells the driver that he/she is sleepy and should take a break.
- We take special care to create a separate thread responsible for calling alarm to ensure that our main program isn't blocked until the sound finishes playing.

5.1.2 ALARM2:

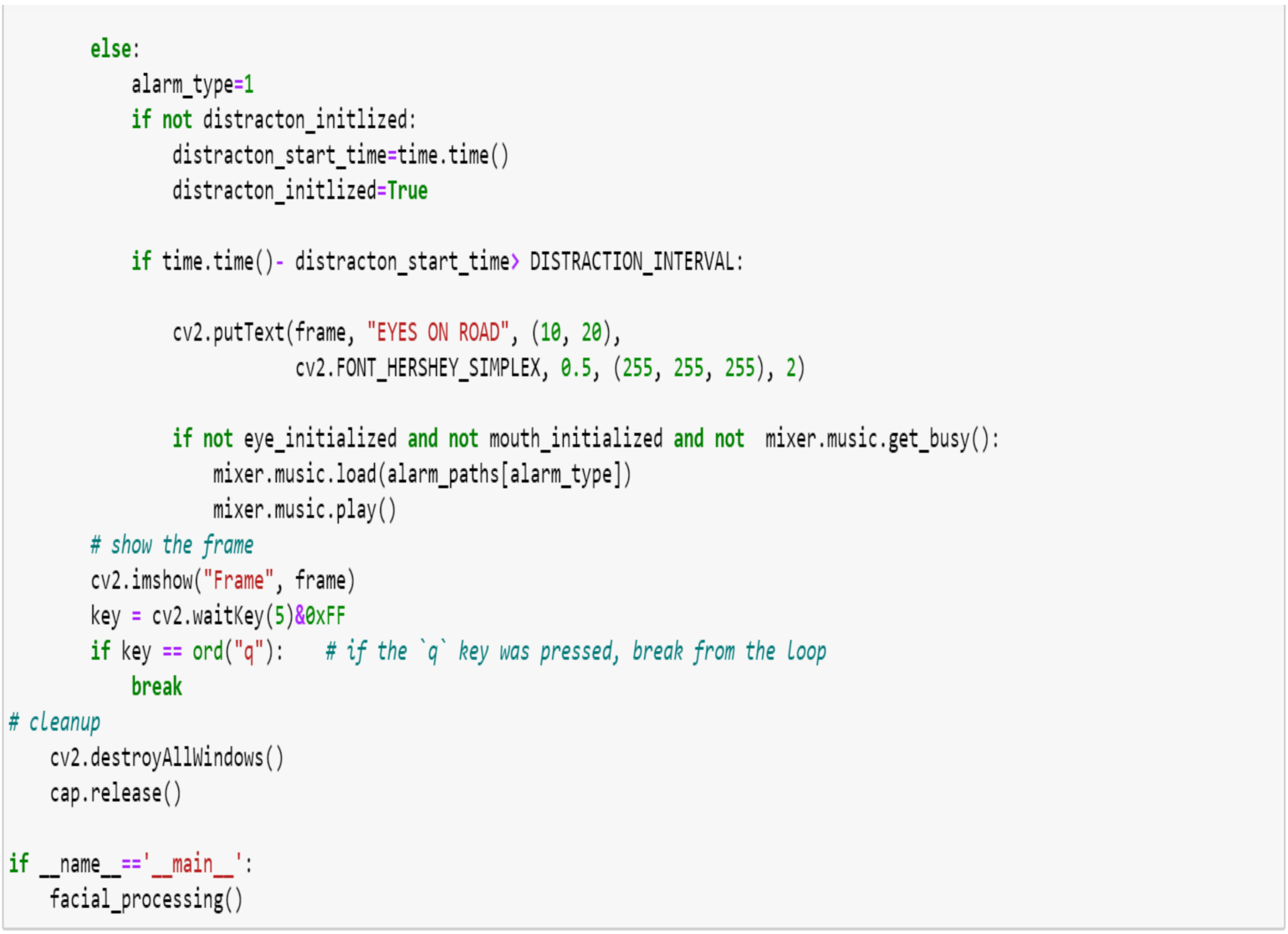


```
# check if the mar is below the MOUTH_DROWSINESS_THRESHOLD

            if mar > MOUTH_DROWSINESS_THRESHOLD:

                if not mouth_initialized:
                    mouth_start_time= time.time()
                    mouth_initialized=True

# if the mouth was opened for a sufficient time then sound the alarm

                if time.time()-mouth_start_time >= MOUTH_DROWSINESS_INTERVAL:
                    alarm_type=0
                    cv2.putText(frame, "YOU ARE YAWNING...\nDO YOU NEED A BREAK?", (10, 40),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

                    if not mixer.music.get_busy():
                        mixer.music.load(alarm_paths[alarm_type])
                        mixer.music.play()
            else:
                mouth_initialized=False
                if not distracton_initlized and not eye_initialized and mixer.music.get_busy():
                    mixer.music.stop()
```

Figure 5.1.2 alarm2

- Our first alarm will be popped when the calculated mouth aspect ratio is greater than the mouth_threshold_ratio.
- The alarm tells the driver that he/she is yawning and should take a break.

5.1.3 ALARM3:



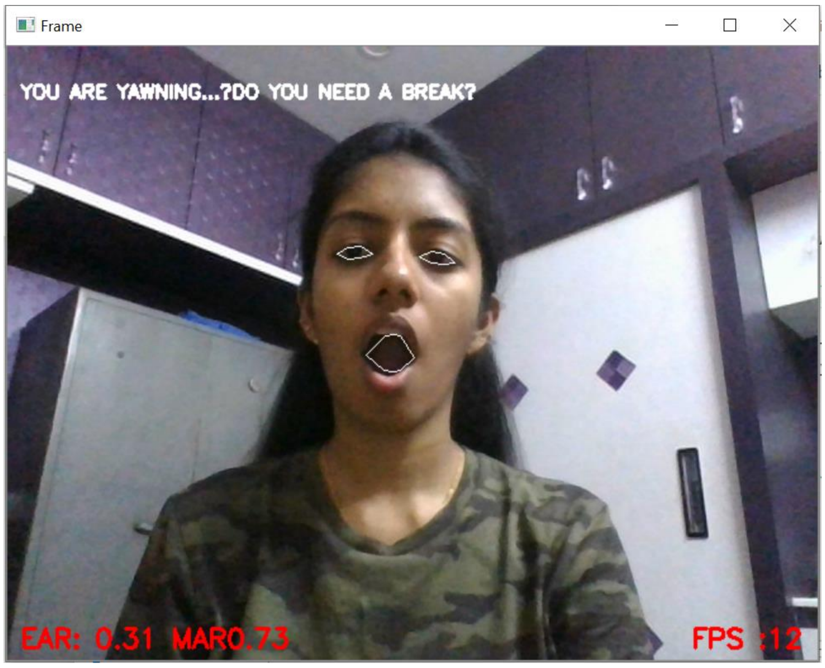
```
        else:
            alarm_type=1
            if not distracton_initlized:
                distracton_start_time=time.time()
                distracton_initlized=True

            if time.time()- distracton_start_time> DISTRACTION_INTERVAL:

                cv2.putText(frame, "EYES ON ROAD", (10, 20),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

                if not eye_initialized and not mouth_initialized and not  mixer.music.get_busy():
                    mixer.music.load(alarm_paths[alarm_type])
                    mixer.music.play()
        # show the frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(5)&0xFF
        if key == ord("q"):    # if the `q` key was pressed, break from the loop
            break
# cleanup
    cv2.destroyAllWindows()
    cap.release()

if __name__=='__main__':
    facial_processing()
```

Figure 5.1.3 alarm3

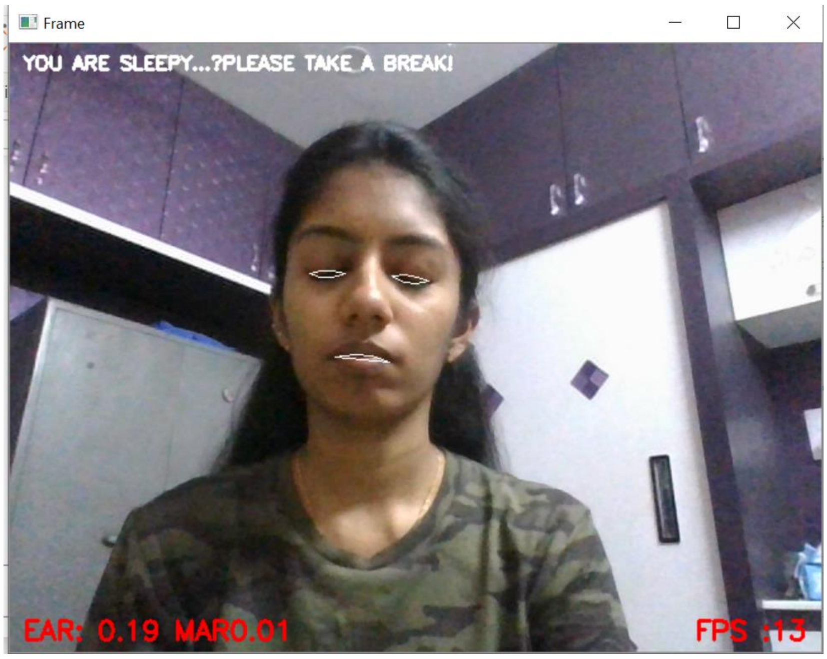
- The final code block in our drowsiness detector handles displaying the output frame to our screen.
- Use the function cv2.imshow() to display an image in a window. The window automatically fits to the image size.
- Cv2.waitKey() is a keyboard binding function. Its argument is the time in milliseconds. The function waits for specified milliseconds for any keyboard event.
- If the ‘q’ is pressed, break from the loop.
- We use cv2.destroyAllWindows() simply destroys all the windows we created.

OUTPUT:

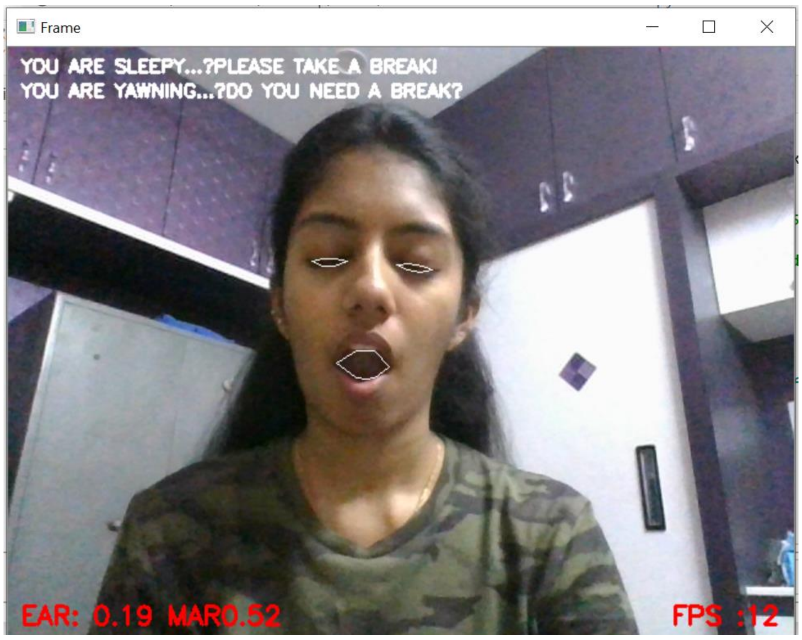
In this case, when eyes are open but yawning is detected in continuous frames, alarm with a message “YOU ARE YAWNING …? DO YOU NEED A BREAK?” is popped.



In this case, when eyes are closed and drowsiness is detected in continuous frames, alarm with a message “YOU ARE SLEEPY …? PLEASE TAKE A BREAK?” is popped.



In this case, when eyes are closed and yawning is detected in continuous frames, alarm with a message “YOU ARE YAWNING …? DO YOU NEED A BREAK?” and “YOU ARE SLEEPY …? PLEASE TAKE A BREAK?” is popped.

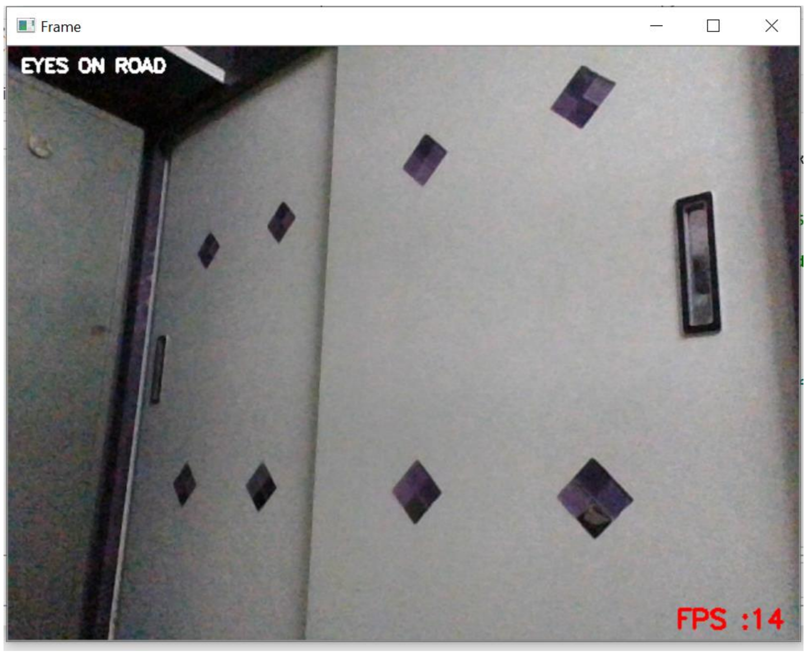


YOU ARE SLEEPY...?PLEASE TAKE A BREAK!
YOU ARE YAWNING...?DO YOU NEED A BREAK?

EAR: 0.19 MAR0.52

FPS :12

In this case, when you are away from the screen in continuous frames, alarm with a message “EYES ON ROAD” is popped.



EYES ON ROAD

FPS :14

+CONCLUSION:

The purpose of our project is to solve one of the most effective problem in our day to day life. To develop a project with low cost which can detect drowsiness. As a result of our project, the ratio of accidents can be reduced. Our proposed system is capable of detecting the sleepiness and distraction of the driver which reduces the concentration of the driver that leads to accidents. So, by developing the project we can reduce the accidents caused due to sleepiness.

REFERENCES:

www.pyimagesearch.com

https://www.pantechsolutions.net/driver-drowsiness-detection-using-opencv-and-python

https://github.com/ruchithasai/AIandML-Project