DROWSINESS DETECTION AND PREVENTING ACCIDENT BY USING K-NEAREST NEIGHBOR(KNN) ALGORITHM

A PROJECT REPORT

Submitted by

M. PANDIESWARI

(Reg. No: 21MCR076)

S. PRIYADHARSHINI

(Reg. No: 21MCR082)

K. SOWNTHAR

(Reg. No: 21MCR101)

in partial fulfilment of the

requirement for the award of the degree

of

MASTER OF COMPUTER APPLICACTIONS

DEPARTMENT OF COMPUTER APPLICATIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE-638 060

JANUARY-2023

DEPARTMENT OF COMPUTER APPLICACTIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE - 638 060

JANUARY-2023

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"DROWSINESS DETECTION AND PREVENTING ACCIDENT BY USING K-NEAREST NEIGHBOR(KNN) ALGORITHM"** is the bonafide record of project work done by **M.PANDIESWARI (Reg.No: 21MCR076), S.PRIYADHARSHINI (Reg.No: 21MCR082), K.SOWNTHAR (Reg.No: 21MCR101)** in partial fulfilment for the award of Degree of Master of Computer Applications of Anna University, Chennai during the year 2022-2023.

SUPERVISOR **HEAD OF THE DEPARTMENT**

(Signature with Seal)

Date:

Submitted for the end semester viva-voce examination held on ________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We affirm that the project report titled **"DROWSINESS DETECTION AND PREVENTING ACCIDENT BY USING K-NEAREST NEIGHBOR(KNN) ALGORITHM"** being submitted in partial fulfillment of the requirements for the award of Master of Computer Applications is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

M.PANDIESWARI
(Reg.No: 21MCR076)

Date:

S.PRIYADHARSHINI
(Reg.No: 21MCR082)

K.SOWNTHAR
(Reg.No: 21MCR101)

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date: Name and Signature of the Supervisor

[Dr.L.RAHUNATHAN]

ABSTRACT

No one can survive without transport now a days. Transportation causes accident not at all time but often. Accident becomes a daily part of our life. It is caused by negligence of a person who drives. We cannot avoid transportation but can reduce the count of accident level. There are various terms cause accident they are due to rash drive, drunk and drive, riskless driving, road rage, potholes, drowsy driving, uncontrolled speed, deadly curves. Behalf of this we are trying to reduce accident by detecting the sleeping state of a person who drives the vehicle.

Drowsiness comes under fatal accidents. Normal person blinks their eyes 20 times per minute but person in sleeping state blinks above 20 times per minute. By detecting the closure of eye, the Eye Aspect Ratio will be calculated. And also Mouth Aspect Ratio is calculated by calculating the count of yawning, opening and closing state of mouth. After calculating both values as separate the average of both will be calculated using specific formula. In this Supervised learning technique of KNN classification algorithm is used to predict the values by using feature similarity method. It gives the output based on closely matches value in compared with all the values. K-Nearest Neighbor (KNN) algorithm is used to ensure the detection of drowsiness in order to avoid accident. Using such system, it is possible to avoid accident by alerting a person from falling sleep.

Detection of drowsiness of a person who drives a vehicle to avoid the accident by calculating Eye Aspect Ratio and Mouth Aspect Ratio using separate formula. And averaging them by using the specific formula of calculating the MOE value. K-Nearest Neighbor [KNN] algorithm is used as classifier to predict the accuracy of the dataset. Jupyter Notebook is the greatest tool for implementing Python programming because it comes with a variety of libraries and header files. Jupyter improves the accuracy and precision of the task.

ACKNOWLEDGEMENT

We respect and thank our correspondent **Thiru.A.K.ILANGO BCom., MBA., LLB.**, and our Principal **Dr.V.BALUSAMY BE(Hons)., MTech., PhD.,** Kongu Engineering College, Perundurai for providing us with the facilities offered.

We convey our gratitude and heartfelt thanks to our Head of the Department **Dr.R.THAMILSELVAN ME., PhD.,** Department of Computer Applications, Kongu Engineering College, for his perfect guidance and support that made this work to be completed successfully.

We also wish to express our gratitude and sincere thanks to our project coordinators **Dr.L.RAHUNATHAN MCA., PhD.,** Associate Professor and **Mrs.S.HEMALATHA MCA.,** Assistant Professor(Sr.G) Department of Computer Applications, Kongu Engineering College, who has motivated us in all aspects for completing the project in the scheduled time.

We would like to express our gratitude and sincere thanks to our project guide **Dr.L.RAHUNATHAN MCA., PhD.,** Associate Professor Department of Computer Applications, Kongu Engineering College for giving his valuable guidance and suggestions which helped us in the successful completion of the project.

We owe a great deal of gratitude to our parents for helping overwhelm in all proceedings. We bow our heart and head with heartfelt thanks to all those who thought us with their warm services to succeed and achieve our work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
EAR	Eye Aspect Ratio
MAR	Mouth Aspect Ratio
MOE	Mouth aspect ratio Over Eye aspect ratio
KNN	K Nearest Neighbor
PUC	Pupil Circularity
ML	Machine Learning
EEG	Electroencephalogram
ECG	Electrocardiogram
EOG	Electrooculogram

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

Drowsiness is common for all the human in such that we are here to predict the drowsy of a person. Faces contain information that can be used to interpret levels of drowsiness. There are many facial features that can be extracted from the face to infer the level of drowsiness. These include eye blinks, head movements and yawning. However, the development of a drowsiness detection system that yields reliable and accurate results is a challenging task as it requires accurate and robust algorithms. A wide range of techniques has been examined to detect driver drowsiness in the past.

The recent rise of machine learning requires that these algorithms be revisited to evaluate their accuracy in detection of drowsiness. As a result, this paper reviews machine learning techniques which include classifier of KNN. The analysis reveals that K Nearest Neighbor technique is the most commonly used technique to detect drowsiness. There are several mathematical calculations are used in identifying the sleepy state of a driver. In search that we are using the following techniques of Eye Aspect Ratio (EAR), Mouth Aspect Ratio (MAR) and combines gives the results which in form known as Mouth over Eye Aspect Ratio.

Machine Learning is one of the efficient technologies for testing, which is based on training and testing. It is the branch of Artificial Intelligence (AI) which is one of the broad areas of learning where machines emulate human abilities, machine learning is a specific branch of AI. On the other hand, machine learning systems are trained to learn how to process and make use of data hence the combination of both techniques. With the help of machine learning techniques, the sleepy state of a driver can be detected in earlier stage of drowsiness. So that the accident can be avoided and it is a step to prevent a person from the accident. Machine learning programs can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

Drowsiness refers to feeling sleepy or tired or unable to keep your eyes open. Drowsiness, also called “excess sleepiness” or “somnolence” can be accompanied by lethargy, weakness, and lack of mental agility. Feeling abnormally sleepy or tired during the day is commonly known as drowsiness. Drowsiness may lead to additional symptoms, such as forgetfulness or falling asleep at inappropriate times. Driver drowsiness detection systems can use cameras, eye tracking sensors and other hardware to monitor visual cues, where drowsiness can be detected through yawning frequency, eye-blinking frequency, eye-gaze movement, head movement and facial expressions.

Drowsiness is classified in terms, and there are specific techniques for detecting it. The below figure 1.1 that represents the various techniques of identifying the drowsiness. The driver drowsiness detection system in automotive vehicle focuses on abnormal behavior exhibited by the driver using a microcontroller, the Raspberry pi single board computer. In the proposed system a non-intrusive driver drowsiness monitoring system has been developed using computer vision techniques.

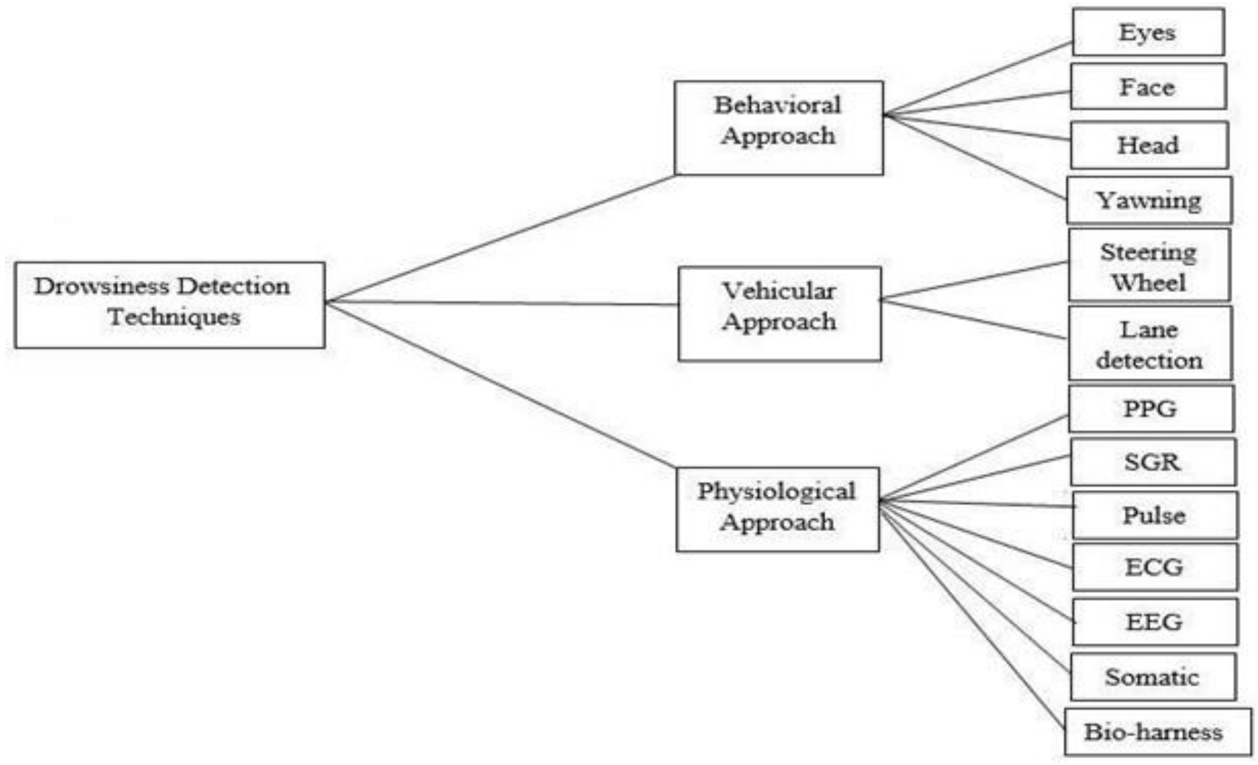


Figure 1.1 Drowsiness Detecting Techniques

1.2 MACHINE LEARNING

Machine learning is an artificial intelligence branch that focuses on developing applications that allow a system to learn and improve accuracy on its own without having to be explicitly programmed. Machine learning is the process of creating computer programs that can access data and learn on their own. The key goal is for computers to be able to learn and adapt without the need for human interaction. As a product of what they've done the two types of machine learning algorithms are unsupervised and supervised machine learning algorithms. Machine learning allows computers to learn and act appropriately. Assists the machine in learning complex models and forecasting data, as well as performing complex math on large datasets. Machine learning-based misinformation detection systems will be reliable. To detect whether the news is fake or real. Machine learning models such as naïve Bayes, decision tree, logistic regression, random forest, and support vector machine are used. Classification is a method for supervised learning. Machine learning algorithms that are supervised can use 3 marked examples to apply what they've learned in the past to new data and forecast future events. Based on the analysis of an established training dataset, the learning algorithm generates an inferred function to make output value predictions. The Machine can do a variety of things. The computer would use an algorithm to analyses the training data and produce a correct result from labelled data. Unsupervised learning is the process of teaching a program to work on data that hasn't been labelled or classified, and then allowing the algorithm to do so without supervision. Classification is a process of categorizing a given set of data into classes, It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label, or categories.

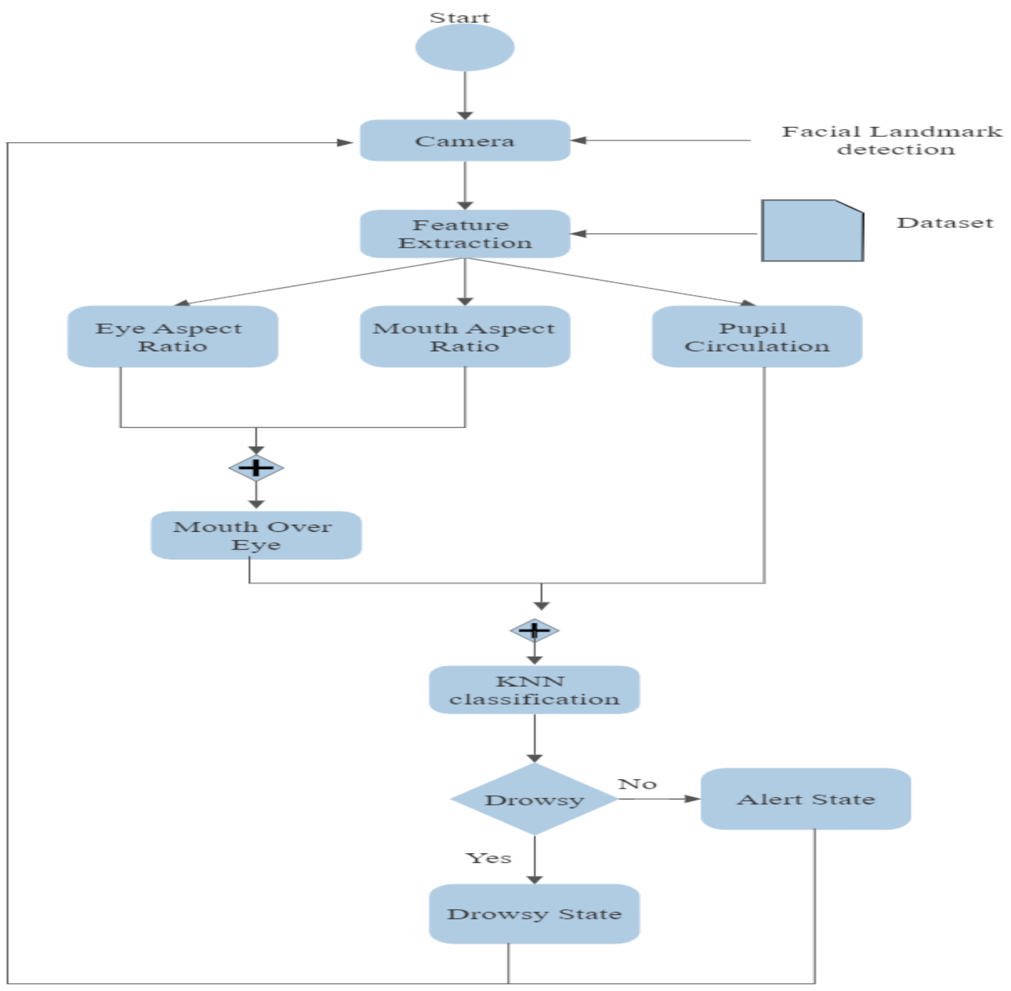


Figure 1.2 System Flow

The Figure 1.2 shows the flow of the system which starts by detecting the face of a person. The images are captured by the camera based on the facial landmarks. Then the dataset moves to the phase of feature extraction which neglects the irrelevant data. The values are getting calculated based on the formula which are specified for EAR, MAR, MOE and PUC. Then the calculated values are implemented in the KNN classifier. By using this algorithm, it is detected as drowsy or not.

SUMMARY

One of the most vital causes of an accident is drowsiness and large number of deaths has been occurred in past year. To reduce that count we are planned to detect the drowsiness of a person who drives a vehicle. So, that the accident can be avoided, it is necessary to predict sleepiness in early stage. Most of the persons are died because of this reason. So, it is important to understand the most effective methods for detecting drowsiness. We are going to calculate the accuracy by using machine learning approach, on the basis of those calculation, to be conclude the best accuracy among them. The literature review of this project is discussed in chapter 2.

CHAPTER 2

LITERATURE REVIEW

Real-Time Classification For Autonomous Drowsiness Detection Using Eye Aspect Ratio[1]

Caio Bezerra et al., [1] proposed by identified the early symptoms of sleeping state and trained by using the different dataset of images and videos for drowsiness detection by using support Vector Machine. CV tools are used to extract and identify signals of abnormal states (i.e., drowsiness) observed in operator images/ videos sensors. The goal is not to detect aspects and status after the failure happens but rather to proactively identify behaviours performed by the operator before it actually takes place avoiding all consequences of the error. Specifically, the detection of eye blinks has an important role in systems that monitor human operator's vigilance. Therefore, an adaptable method that can automatically detect drowsiness directly from an ordinary webcam would be valuable. To evaluate blinks, we used a simple, yet robust metric called eye aspect ratio (EAR) that does not require intensive signal and computer processing (such as EEG and EOG). Indeed, for the EAR evaluation, the proportion between height and width of the eye is calculated (i.e., higher values for open eyes and smaller for closed eyes). This process easily extracts a single feature from the entire image and reduces the amount of computational processing and storage memory required.

Drowsiness Detection System using Eye Aspect Ratio Technique [2]

Saravanaraj et al., [2] suggested by using the EAR technique. The future development can be done by detecting the state of yawning and distraction of the driver by using electrocardiogram (ECG), electroencephalogram (EEG), electrooculogram (EOG) and electromyogram (EMG). One of the technologies is based on the vehicle like autonomous driving that can monitor steering wheel direction, keep the car in lane position and pressure on the accelerator continuous controlling by the engine control unit (ECU), however this is not the most accurate solution for this problem. On the other hands, the physiological of the driver, which continuously measure the driver heart rate and brain activity by electrocardiogram (ECG), electroencephalogram (EEG), electrooculogram (EOG) and electromyogram (EMG) using a particular custom device was invested, up till now it is an impracticable solution.

The role of the system is to detect eyes location from images and calculate the value of EAR. In this method each eye is labelled with 6 (x, y) coordinates in landmarks retuned Dlib predictor function. The labelling is starting at the left-corner of the attention, then working clockwise round the remainder of the region. Meanwhile, there's a relation between the distance of those coordinates. Thus, it derives an equation for this relation called the attention ratio and also known as Eye Aspect Ratio (EAR). According to the experimental results, it successfully detects person during drowsy condition.

Using Eye Aspect Ratio to Enhance Fast and Objective Assessment of Facial Paralysis[3]

Jialing Feng et al., [3] discussed about detecting drowsiness in paralysis patients by calculating their EAR value. 87 paralysis patients are detected in this paper by using ensemble of regression tree (ERT) algorithm, gradient boosting decision tree (GBDT) algorithm. Facial paralysis can lead to the loss of autonomic motor function of the unilateral mimetic muscles of the face. Accurate and objective evaluation of the degree of facial nerve motor function damage is key for the treatment and rehabilitation of facial paralysis patients. Clinically, the House–Brackman grading system (HBGS) is used to evaluate and grade facial nerve motor dysfunction in patients with facial paralysis. Because the description of symptoms of adjacent facial paralysis grades in HBGS is ambiguous, the evaluation is prone to subjective differences. Moreover, HBGS cannot reflect motor dysfunction in the local regions of the eyebrow, eye, mouth, and nose, which affects the design of follow-up treatment and rehabilitation regimen for facial paralysis patients. Therefore, this study trained a new feature point detection model to improve the accuracy of feature point detection and proposed a regional index to quantify the motion symmetry of facial regions. In this study, face image data of facial paralysis patients were used as a training set, and a facial landmark detection model for facial paralysis patients was constructed to calculate the EAR.

The incorrect detection rate of this model for the eye region of facial paralysis patients was 17.1%, an improvement over currently available detection models. Detection and calculation using the constructed detection model yielded the difference in bilateral EAR of 87 facial paralysis patients. Subsequently, the relationship between the FNGS 2.0 score and the corresponding bilateral EAR difference was analysed, and the correlation analysis results showed that the two were highly correlated, with a correlation coefficient of 0.9673. By implementing the proposed method and related experiments, this study proves that the bilateral EAR difference can be applied to the objective and rapid assessment of the severity of facial paralysis.

State Of The Art Analysis Of Modern Drowsiness Detection Algorithm Based On Computer Version [4]

Fudail Hasan et al., [4] described by detecting the yawning state of a person they themselves created a dataset and trained and also blinking state also calculated by using Deep Neural Network. Drive Safely system is aimed at dangerous states' detection based on the camera and the sensors of the smartphone. For this system, drowsiness is one of the dangerous states that should be detected in the vehicle cabin. Initially, we used the Dlib framework to determine facial landmarks. Based on landmark points we detected the opened or closed eyes and opened or closed mouth. Based on this information we detect drowsiness dangerous state. However, when we collected enough data of drivers' faces, we decided to train models that detect eye closeness as well as yawning detection instead of using Dlib framework. Such models are more reasonable than Dlib since it provides better accuracy as well as less computational time. The paper proposes a three-step method for yawning detection.

The first step is to capture the frame from the camera, the second step is to use a face-detector with high accuracy to detect the face of the driver within the frame, and the third step is to classify the sub-image around the face (yawny face or not). The paper also contains a description of the available yawning detection datasets as well as our own dataset that we collected. Such systems should detect the drowsy driver as well as notify him about this situation. Using both, online available dataset and our dataset, we trained the yawning detection model which is a modified version of the "MobileNetV2" classification model. The modified model has a lower accuracy than the original model but it is smaller and faster. In general, the new method increased the accuracy of drowsiness detection.

Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application [5]

Rateb Jabber et al,. [5] revealed EEG with Inertial Measurements Units (IMU) sensors are used to detect the 5 levels of drowsiness. Android Application is used to take the sleepy pictures of the driver by using Convolutional Neural Network (CNN). There has been a significant rise in the number of vehicles that are equipped with Android Auto or Apple Car nowadays. Most of the cars that are being introduced now have these components built-in. Such features are now readily available in lower-end cars too. Due to this, drowsiness detection systems can be easily developed around such built-in Android and iOS platforms. Embedded devices or mobile phones that can easily pair with these car dashboards can be used to enhance driver behaviour detection using simple camera setup and state of the art computer vision systems powered by Deep Learning.

Microsleeps are of short duration where the driver has his eyes closed and is not perceiving any visual information and thereby cannot react if the car departs from the lane or if the car comes to halt due to harsh braking. With the advent arXiv:2002.03728v1. 17 Jan 2020 of new sensors and radars in high-end cars, cars can notify or even take action to avoid such crashes. Although this is a great advancement, it would also be beneficial if the car knows that the driver is sleepy and requests him to take rest. Also, another common issue with the current machine vision models is the fact that most of these algorithms are bulky (large in size) and require dedicated hardware to run the models that have been developed. They do not run efficiently on devices with low computational power. This paper details such a Deep Learning-based drowsy detection algorithm that can potentially enable such an intervention, Moreover, it is simple and easy to configure on any mobile or embedded device. Among the many crash aversion techniques and other safety features that are developed in newer cars, there is a lot of interest on detecting fatigued and drowsy drivers by using cameras, sensors and other instruments to alert and thereby avert fatal crashes.

Driver Drowsiness Detection using Machine Learning Approach [6]

Novie Theresia et al., [6] suggested the Integral image is used for fast feature evaluation. AdaBoost is used to select the small number of features. Then face is detected. All these contributes in detection of drowsiness by using HAAR face detection algorithm, AdaBoost algorithm. Frontal face detection and eye detection respectively are detected in the initial stage. There are three main contributions of the object detection framework have been used to achieve high frame rates. The first contribution of this paper, is a new image representation called an integral image that allows for very fast feature evaluation. It use a set of features which are reminiscent of Haar Basis functions In order to compute these features very rapidly at many scales, the integral image representation for images is introduced. The proposed method to calculate M-EAR can determine blinking eyes and know whether the driver is sleepy or not.

Using a few operations per pixel, the integral image can be computed from an image. The second contribution is a method for constructing a classifier by selecting a small number of important features using AdaBoost. In order to ensure rapid classification, the learning process must exclude a large majority of the available features, and focus on a small set of essential features. This feature selection is achieved through a simple modification of the AdaBoost procedure. The third contribution of this paper is a method for combining successively more complex classifiers in a cascade structure which dramatically rapids the speed of the detector by focusing attention on promising regions of the image. This object detection procedure classifies images based on the value of simple features. More specifically it is based on three kinds of features. A three-rectangle feature computes the sum within two outside rectangles subtracted from the sum in a centre rectangle.

Driver Safety Development: Real Time Driver Drowsiness Detection System Based on Convolutional Neural Network [7]

Maryam Hashemi et al., [7] proposed Eye detection and pre-processing done then collection process is used in detection of drowsiness. Time cost for eye closure detection in different methods by using Fully Designed Neural Network (FD-NN). This paper focuses on the challenge of driver safety on the road and presents a novel system for driver drowsiness detection. In this system, to detect the falling sleep state of the driver as the sign of drowsiness, Convolutional Neural Networks (CNN) are used with regarding the two goals of real-time application, including high accuracy and fastness. Three networks introduced as a potential network for eye status classification in which one of them is a Fully Designed Neural Network (FD-NN), and others use Transfer Learning in VGG16 and VGG19 with extra designed layers (TL-VGG).

There are three main categories of drowsiness detectors: Vehicle-based, Signal-based, and Facial feature-based. Vehicle-based methods try to infer drowsiness from vehicle situation and monitor the variations of steering wheel angle, acceleration, lateral position, etc. The system detects drowsiness in approximately less than 20 seconds after the subject starts driving and also maximize the data accuracy at over 90%. Section III details the algorithm that is being employed followed by the conclusion which is the final section. However, these approaches are too slow for real-time tasks. Signal-based methods infer drowsiness from psychophysiological parameters. Several studies have been done during the last years based on these methods.

Real Time Driver Drowsiness Detection Using a Logistic-Regression Based Machine Learning Algorithm [8]

Mohsen Babaeian et al., [8] discussed by using ECG sensor is used to detect the electrical signal from the heart. Due to irregular pattern of heart resamples the uniformity sampled signal by using Naïve Bayes Method, logistic regression, In order to detect drowsiness in early stages, there have been many methods developed with the help of sensors mostly associated with many different physiological signals such as electrocardiogram (ECG) and others. Current research included studies that record and analyse a driver's blinking pattern and studies that monitor brain electrical behaviour to detect driver drowsiness. In our research, the focus is on ECG signal variation in transition to drowsiness and detection of the changing ECG signal behaviour to accomplish high accuracy in less time. In order to implement a drowsiness detection system, a hardware and software system capable of detecting electrical signals generated by the heart in real time and predicting the states of drowsiness needs to be designed. Goal is to detect drowsiness in a few seconds, after the driver starts driving, while maintaining a level of accuracy above 90%.

Real-Time Driver Drowsiness Detection System Using Eye Aspect Ratio and Eye Closure Ratio [9]

Sukrit Mehta et al., [9] proposed approach to detect driver's drowsiness that works on two levels. The application is installed on driver's device running Android operating system (OS). The process starts with capturing of live images from camera and is subsequently sent at local server. At the server's side, Dlib library is employed to detect facial landmarks and a threshold value is used to detect whether driver is drowsy or not. These facial landmarks are then used to compute the EAR and are returned back to the driver. In our context, the EAR value received at the application's end would be compared with the threshold value taken as 0.25 If the EAR value is less than the threshold value, then this would indicate a state of fatigue. In case of Drowsiness, the driver and the passengers would be alerted by an alarm.

When the eyes are open, the numerator value increases, thus increasing the EAR value, and when the eyes are closed the numerator value decreases, thus decreasing the EAR value. In this context, EAR values are used to detect driver's drowsiness. The above condition implies that the person has closed his/her eyes and is in a drowsy state. On the contrary, if the EAR value increases again, it implies that the person has just blinked the eye and there is no case of drowsiness.

Drowsiness Detection According to the Number of Blinking Eyes Specified From Eye Aspect Ratio Value Modification [10]

Joseph Felix et al., [10] suggested the research on sleepiness in motorists has been done by many researchers. Several stages of drowsiness detection: the first face detection and eye detection with Viola and Jones algorithm, Histogram of oriented gradient (HOG) detector, and Convolutional Neural Network(CNN). The next process is feature extraction using: Landmarks, and Local Binary Pattern (LBP). The features obtained were analyzed using percentage of eye closure (PERCLOS), blink frequency, eye aspect ratio (EAR), face position, nodding frequency. The last classified with Support Vector Machines (SVM), KNN, and Hidden Markov Model (HMM) to detect sleepiness of the driver.

Drivers are said to be sleepy if eyes are closed more than 45 frames and blinking eyes below 10 per minute. It can be concluded that driver 2 and driver 5 are not sleepy because the number of blinking is more than 10 blinking per minute, while drivers 1 and 3 are not sleepy but are tired because the number of blinking between 20 to 40, and driver 4 can be said to be drowsy because the amount of blinking is below 10 per minute The proposed method to calculate M-EAR can determine blinking eyes and know whether the driver is sleepy or not. From the experimental data of five drivers (each of five videos), then based on the reference that the condition is sleepy if the number of blinks below 10 per minute value, it can be determined fourth driver in sleepy condition.

Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State[11]

Venkata Rami Reddy Chirra et al., [11] proposed that Driver drowsiness is one of the reasons for large number of road accidents these days. With the advancement in Computer Vision technologies, smart/intelligent cameras are developed to identify drowsiness in drivers, thereby alerting drivers which in turn reduce accidents when they are in fatigue. In this work, a new framework is proposed using deep learning to detect driver drowsiness based on Eye state while driving the vehicle. To detect the face and extract the eye region from the face images, Viola-Jones face detection algorithm is used in this work. Stacked deep convolution neural network is developed to extract features from dynamically identified key frames from camera sequences and used for learning phase. A SoftMax layer in CNN classifier is used to classify the driver as sleep or non-sleep. This system alerts driver with an alarm when the driver is in sleepy mood. The proposed work is evaluated on a collected dataset and shows better accuracy with 96.42% when compared with traditional CNN. The limitation of traditional CNN such as pose accuracy in regression is overcome with the proposed Staked Deep CNN.

In this proposed work a new method is proposed for driver drowsiness detection based on eye state. This determines the state of the eye that is drowsy or non- drowsy and alert with an alarm when state of the eye is drowsy. Face and eye region are detected using Viola-Jones detection algorithm. Stacked deep convolution neural network is developed to extract features and used for learning phase. A SoftMax layer in CNN classifier is used to classify the driver as sleep or non-sleep. Proposed system achieved 96.42% accuracy. Proposed system effectively identifies the state of driver and alert with an alarm when the model predicts drowsy output state continuously. In future we will use transfer learning to improve the performance of the system.

Drowsiness Detection Based on Eye Closure and Yawning Detection[12]

Sheela Rani et al., [12] presented by Haar based classifier is used in extraction of facial features to detect the eye closure and yawning state. Alarm will be sounded when the eye closure and yawing detected by using Haar Based Classifier, Support Vector Machine (SVM), Convolutional Neural Network (CNN) in the tittle of Drowsiness Detection Based on Eye Closure and Yawning Detection. Based on yawning measurement. This involves several steps including the real time detection and tracking of driver’s face, detection and tracking of the mouth contour and the detection of yawning based on measuring both the rate and the amount of changes in the mouth contour area. APEX™ automotive smart camera platform developed by Connive Corp. In our approach, the driver’s face is continuously captured using a video camera that is installed under the front mirror inside the car, Next, detecting drowsiness involves two main steps to properly measure changes in facial gestures that imply drowsiness.

SUMMARY

Here by, we have referred 12 papers are regarding the Drowsiness detection, in that most authors of the paper focused on Machine learning techniques but some of the papers also focused on deep learning concepts and used various algorithms in detection of drowsiness of a person while driving the vehicle. Based on the various papers, the proposed work will be declared in chapter 3.

CHAPTER 3

PROPOSED WORK

3.1 PROPOSED METHODOLOGY

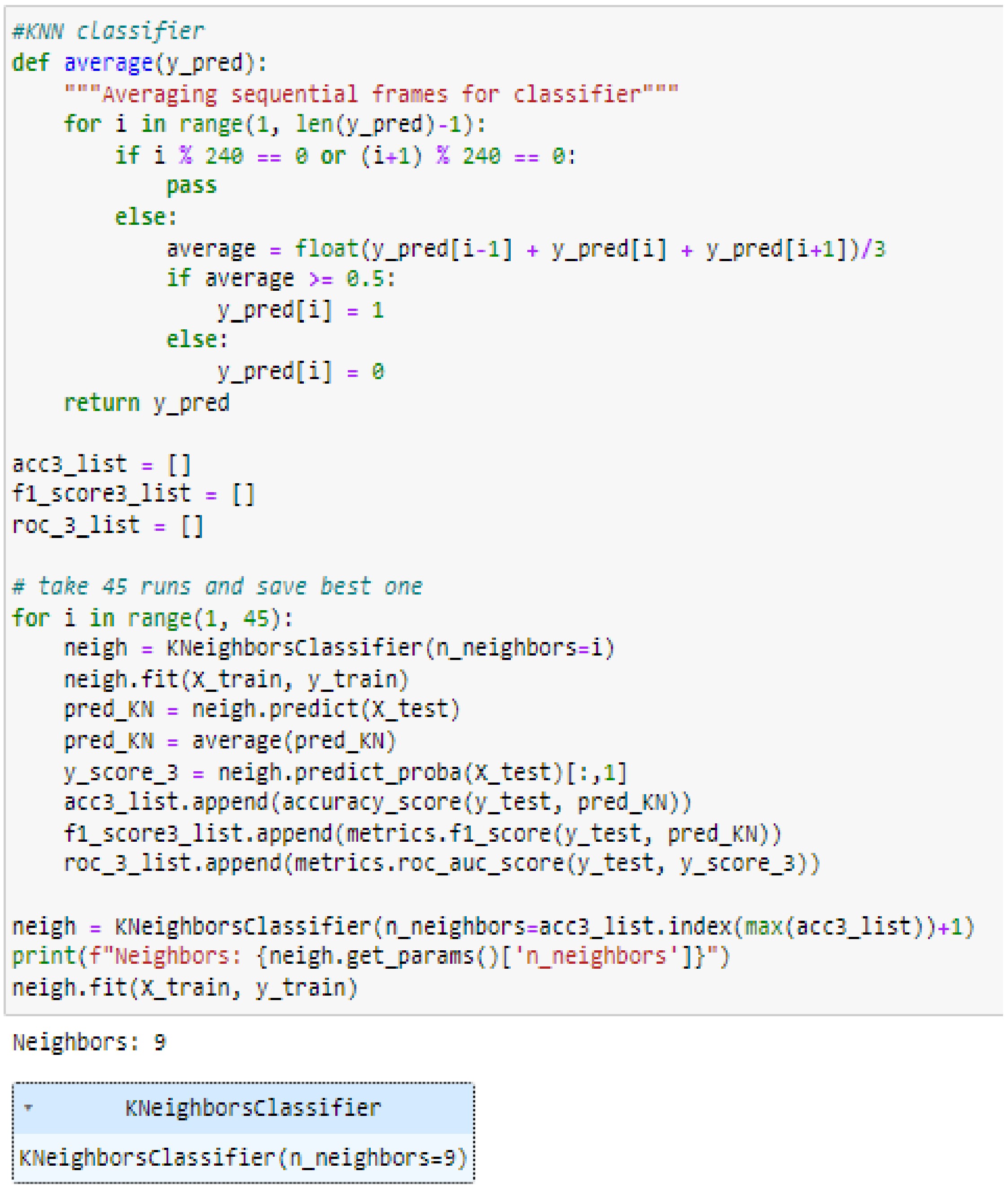
3.1.1 Machine Learning classifiers

In the proposed approach machine learning algorithm has used for predicting the drowsiness. Machine learning algorithms are being used all over the place. The majority of ML algorithms are based on mathematical operations. When the input dataset includes numerical values, implementing an algorithm is a simple operation. However, if the dataset includes categorical data, certain transformations must be performed before implementing machine learning algorithms. Since the dataset for predicting drowsiness includes numerical data. Good results can be obtained by analyzing the main properties of a dataset with the model that is best suited for that problem. A machine learning algorithm has been used for identifying the drowsiness of a person who drives a vehicle. In the above Figure 3.1 represents Machine learning classifiers. To predict drowsiness, tested data has been applied to the machine learning algorithms as K Nearest Neighbor.

3.1.1.1 K-Nearest Neighbor Algorithm

One of the important fundamental Machine Learning tasks is the KNN. KNN is based on the technique of Supervised Learning. The KNN method takes that the new data and previous cases are similar and places the new case in the most similar category to the existing 14 categories. The KNN algorithm saves all possible data and assorts fresh data points according to their similarity.

The below Figure 3.1 represents the Machine Learning classifiers which is used in prediction of drowsiness of a person while travelling.



```
#KNN classifier
def average(y_pred):
    """Averaging sequential frames for classifier"""
    for i in range(1, len(y_pred)-1):
        if i % 240 == 0 or (i+1) % 240 == 0:
            pass
        else:
            average = float(y_pred[i-1] + y_pred[i] + y_pred[i+1])/3
            if average >= 0.5:
                y_pred[i] = 1
            else:
                y_pred[i] = 0
    return y_pred

acc3_list = []
f1_score3_list = []
roc_3_list = []

# take 45 runs and save best one
for i in range(1, 45):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train, y_train)
    pred_KN = neigh.predict(X_test)
    pred_KN = average(pred_KN)
    y_score_3 = neigh.predict_proba(X_test)[:,1]
    acc3_list.append(accuracy_score(y_test, pred_KN))
    f1_score3_list.append(metrics.f1_score(y_test, pred_KN))
    roc_3_list.append(metrics.roc_auc_score(y_test, y_score_3))

neigh = KNeighborsClassifier(n_neighbors=acc3_list.index(max(acc3_list))+1)
print(f"Neighbors: {neigh.get_params()['n_neighbors']}")
neigh.fit(X_train, y_train)
```

```
Neighbors: 9
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=9)
```

Figure 3.1 K-Nearest Neighbor (KNN) Algorithm

Figure 3.1 explains that flow of implementation of the K Nearest Neighbor algorithm. By calculating the average value of MOE and PUC, it starts the implementing process. To predict the class of Machine learning KNN individual based on multiple predictor variables you need to use K Neighbor classifier(). To adjust data values you need to pass X_train and Y_train as parameters in the fit(). To predict the data values based on the trained data module you need to use predict() and you need to pass the x_test values as a parameter in the predict(). To predict accuracy, you must use accuracy score() and pass the predictive data value and y_test values as parameters in the function. The value of k has been taken as 9.

3.2 WORKFLOW DESCRIPTION

3.2.1 Dataset Collection

The dataset used for the detection of drowsiness is prepared by capturing the image of person. Then the captured images are saved as a csv file and so it can be loaded in easy way while implementing. The dataset had been split into two categories firstly a training set and secondary testing set.

3.2.2 Data Preprocessing

In this stage, the given datasets were pre-processed to remove the noise such as punctuation marks, stop words, HTML tags, url, emojis, etc. Pre-processing was done using the NLTK toolkit which is an open-source and broadly used NLP library. It has built-in functions and algorithms such as nltk.

3.2.3 Feature Extraction

Feature extraction is a dimensionality reduction procedure that reduces an initial collection of raw data to more manageable groups for processing. The enormous number of variables in these large data sets necessitates a large number of computational resources to process. Feature extraction refers to approaches that choose and/or combine variables to form features, hence minimizing the quantity of data that must be processed while properly and thoroughly describing the original data set.

3.2.4 Training the classifier

As briefly alluded to earlier, based on the facial landmarks that we extracted from the frames of the videos, we ventured into developing suitable features for our classification model. While we hypothesized and tested several features, the four core features that we concluded on for our final models were eye aspect ratio, mouth aspect ratio, pupil circularity, and finally, mouth aspect ratio over eye aspect ratio.

3.2.5 Feature Normalization

When we were testing our models with the four core features discussed above, we witnessed an alarming pattern. Whenever we randomly split the frames in our training and test, our model would yield results with accuracy as high 70%, however, whenever we split the frames by individuals, our model performance would be poor as alluded to earlier. This led us to the realization that our model was struggling with new faces and the primary reason for this struggle was the fact that each individual has different core features in their default alert state. That is, person A may naturally have much smaller eyes than person B. If a model is trained on person B, the model, when tested on person A, will always predict the state as drowsy because it will detect a fall in EAR and PUC and a rise in MOE even though person A was alert. Based on this discovery, we hypothesized that normalizing the features for each individual is likely to yield better results and as it turned out, we were correct. To normalize the features of each individual, we took the first three frames for each individual's alert video and used them as the baseline for normalization. The mean and standard deviation of each feature for these three frames were calculated and used to normalize each feature individually for each participant.

3.2.6 Result Classification

By implementing all the above steps further, we move to the classification of the result whether the obtained result. On the basis of various ML techniques, we conclude the output of the generated result. target denotes 0 as well as denotes 1. Finally, we determine how much accuracy attend by the implemented ML algorithm.

Here first of all collection of the relevant dataset is required, following that step data pre-processing and cleaning are going to be done using various methodology like Tokenizing, steaming, Stop word removal, etc. Then further move to the next step and feature extraction will be done. Once the feature extraction is done in the next stage dataset is going to be trained and tested. After implementation of all previous steps at the last result classification will happen if the result got as 0 which means it is target similarly if the result got as 1 that shows that it is true. For all the above steps and procedure different machine learning techniques is used. Every machine learning technique will give a different result and will compare the entire machine learning algorithm and move forward with the K Nearest Neighbor classifier gives the best accuracy obtained.

SUMMARY

In this proposed work, machine learning algorithm which is KNN, it is used to identify the drowsiness detection in person. Based on the accuracy attended by the algorithm is obtained that K Nearest Neighbor classifier which performs well and get the highest accuracy of 90.74 %. Based on the proposed work, project implementation is explained in chapter 4.

CHAPTER 4

PROJECT IMPLEMENTATION

4.1 MATHEMATICAL CALCULATION

There are several mathematical terms used to identify the drowsy of a person. Compared to all those terms we are using some of them which are used in creating of features.

4.1.1. FACIAL LANDMARKS

In face there are totally 68 landmarks per frame. It is a task which uses in prediction of points in facial regions includes eyes, nose, lips, mouth, eye brows. Every region in face has a certain range of points. Including all we are going to focusing on eye and mouth which has the points in the range of 37-68. The fixed facial landmarks are pointed and shown in the Figure 4.1.

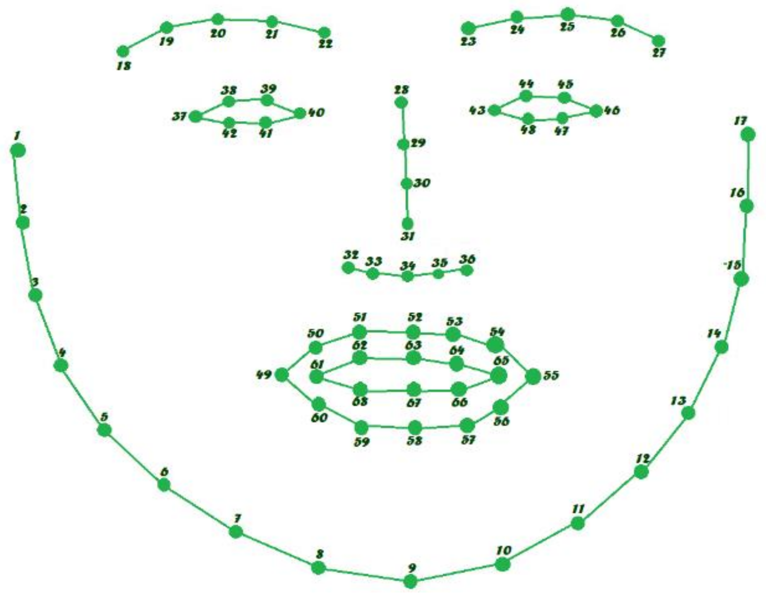


Figure 4.1 Landmark Points

4.1.2 EYE ASPECT RATIO (EAR)

Drowsiness in a person is easily identified by seeing a person eyes. Based on this concept a ratio value of eye is going to be calculate by using the mathematical formula. Length of eyes is calculated by finding and averaging of two distinct vertical lines across the eyes. Each eye has the fixed value of 6 landmarks. Eyes will get wrinkled during drowsy state, this state of eye would be captured and the value of EAR will be calculated.

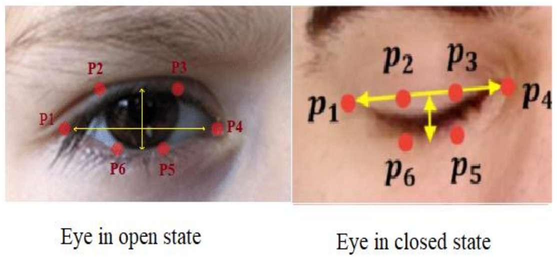


Figure 4.2 Eye in various States with Landmark

$$\text{EAR} = \frac{||p2-p6||+||p3-p5||}{2||p1-p4||}$$

4.1.3 MOUTH ASPECT RATIO (MAR)

Yawning is one of the symptoms of sleepiness. Based on this term, the MAR is computed. It is majorly focused on length and width of the mouth. Opening and closing state of mouth has been detected by using landmarks in mouth then MAR will be formulated. Facial points range from 49 to 68 which represents in the mouth co-ordinates. In the calculation of MAR value only 8 points are needed and their ranges of 61 to 68. MAR value will be higher than the usual level of state.

$$\text{MAR}= \frac{||p2-p8||+ ||p3-p7||+||P4-P6||}{2||p1-p5||}$$

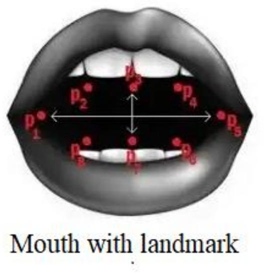


Figure 4.3 Landmarks Points in Mouth

4.1.4. MOUTH ASPECT RATIO OVER EYE ASPECT RATIO (MOE)

A combination ratio value of both MAR and EAR gives the results as mouth over eye ratio value. Where eye ratio value and mouth ratio value both are moving in opposite direction of individual state of changes. Numerator and denominator moves in the opposite directions. So, mouth over eye ratio value increased compared to other features when the person felt drowsy.

$$MOE = \frac{MAR}{EAR}$$

4.1.5. PUPIL CIRCULARITY (PUC)

The Pupil Circularity (PUC) extends from the ratio value of eye. Rather calculating ratio of whole eye counting the changes in pupil can be more effective. Comparing to eye ratio value PUC value gives a great significance in detecting drowsy of a person.

Formula used in calculation of Perimeter

$$\text{PERIMETER} = \text{Distance}(p1,p2)+\text{Distance}(p2,p3)+\text{Distance}(p3,p4)+ \text{Distance}(p4,p5)+\text{Distance}(p5,p6)+\text{Distance}(p6,p1)$$

Formula used in calculation of Area

$$\text{AREA} = \left(\frac{\text{Distance}(p2,p5)}{2}\right)^2 * \pi$$

Formula used in calculation of Circularity

$$\text{CIRCULARITY} = \frac{4 * \pi * \text{Area}}{\text{Perimeter}^2}$$

4.2 KNN CLASSIFIER

Supervised learning classifier of K- Nearest Neighbors algorithm is used to predict the accuracy in high level. Where KNN is non-parametric classifier, it can use for all supervised problems. It is used by grouping up of data point individually. Here we have taken average prediction value of K is 9. Using this k value the highest accuracy is going to be predicted.

Step 1: Import the library predefined functions which are required in implementation process.

Step 2: For implementing we first need the dataset, load the dataset which is saved in the form of csv file

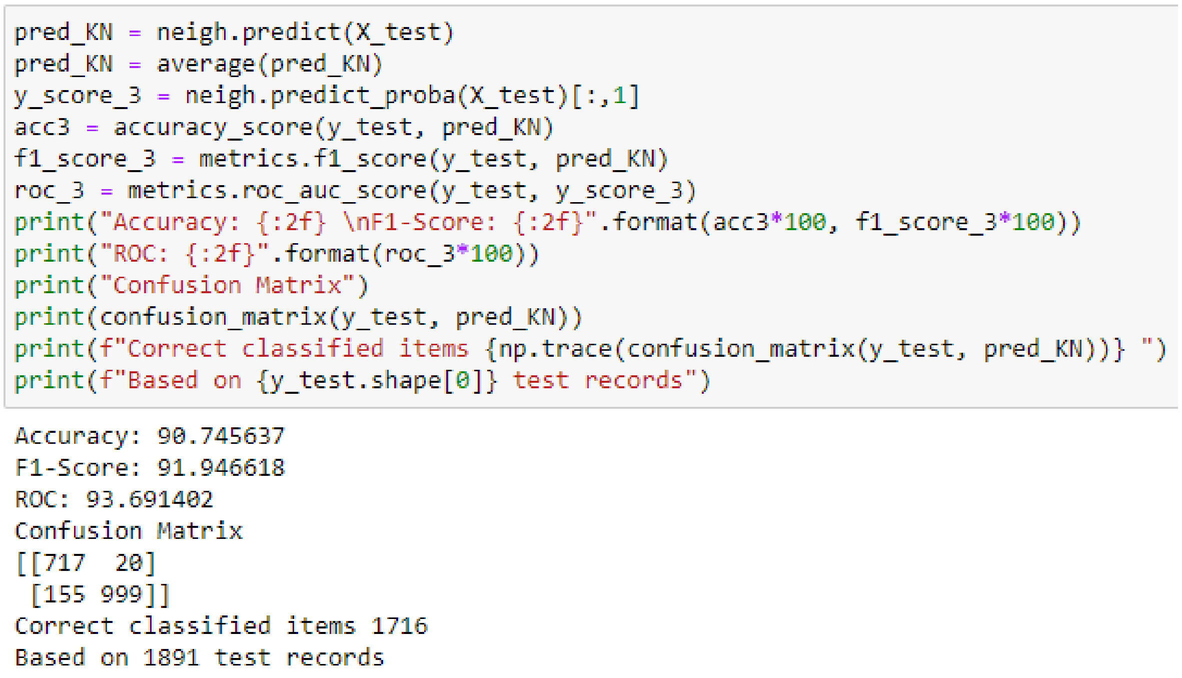
Step 3: Separate the dataset in two types as training data and testing data in the features of x and y points.

Step 4: 80% of data for training as x and 20% of testing data as y has been parted.

Step 5: To choose the k value the distance between each data points should be calculated.

Step 6: By using the method of Euclidean the k value is going to be determined.

Step 7: Here the nearest value of k is 9, by keeping the value of k=9 the model retrieved its data score.



```
pred_KN = neigh.predict(X_test)
pred_KN = average(pred_KN)
y_score_3 = neigh.predict_proba(X_test)[:,1]
acc3 = accuracy_score(y_test, pred_KN)
f1_score_3 = metrics.f1_score(y_test, pred_KN)
roc_3 = metrics.roc_auc_score(y_test, y_score_3)
print("Accuracy: {:2f} \nF1-Score: {:2f}".format(acc3*100, f1_score_3*100))
print("ROC: {:2f}".format(roc_3*100))
print("Confusion Matrix")
print(confusion_matrix(y_test, pred_KN))
print(f"Correct classified items {np.trace(confusion_matrix(y_test, pred_KN))} ")
print(f"Based on {y_test.shape[0]} test records")
```

```
Accuracy: 90.745637
F1-Score: 91.946618
ROC: 93.691402
Confusion Matrix
[[717  20]
 [155 999]]
Correct classified items 1716
Based on 1891 test records
```

Figure 4.4 Graphical Representation of MOE, PUC

The above Figure 4.4 represents the graphical view of Mouth Over Eye Aspect Ratio value and the Pupil Circularity value. Here by the accuracy has been predicted in basis of averaging the calculated values.

SUMMARY

There are various procedures are followed for the implementation of the project for attending most accurate result. While using Machine Learning algorithms, the proposed work is focused on the terms like Dataset collection followed by Data pre-processing and data cleaning. Once data pre-processing is done Feature Extraction and training the classification work will be done. After implementation of all the above steps the system predicts actual classification.

CHAPTER 5

EXPERIMENTAL RESULTS

Dataset had created by capturing the image through live demo and images are saved in the form of CSV file. Then captured images are labled in the basis of three classes. Where value of 0 is labeled for alert, value of 5 is assigned for low cautious and 10 value is labeled for drowsy. These labels are given by the participants which is based on their state while capturing the image. Using facial landmarks the calculations of EAR, MAR, MOE and PUC. Based on these range of calculation drowsy has been detected in a person who drives a car. All the ratio values are classified through KNN algorithm to predict the accuracy value.

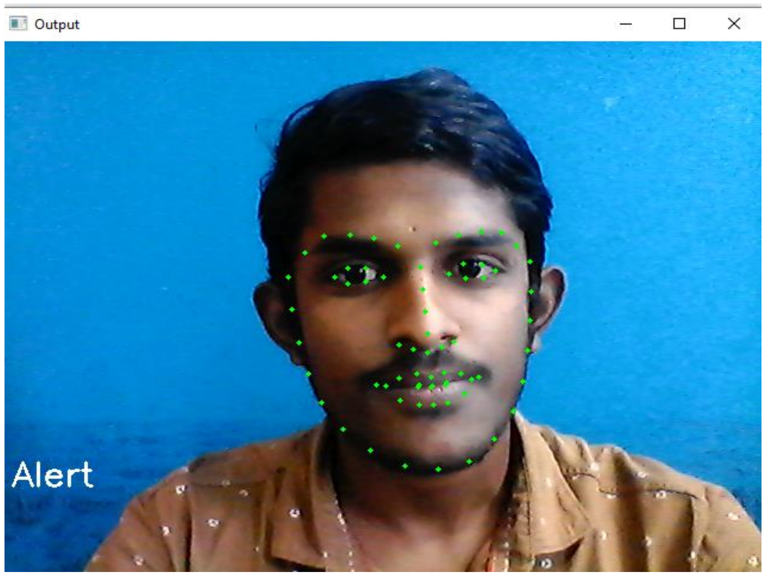


Figure 5.1 Person in a Normal State

The above Figure 5.1 represents how the notification displays when the person is in the normal state. And the alert notification will be displayed.

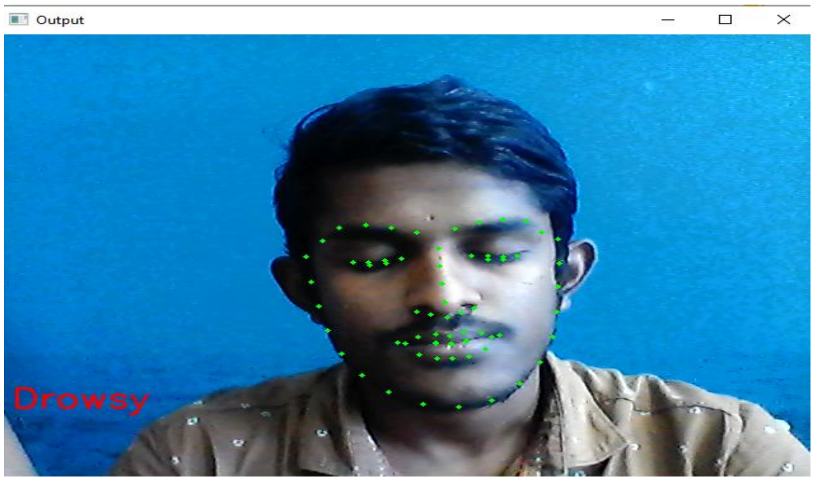


Figure 5.2 Person in Drowsy State

The above Figure 5.2 represents the notification which will be displayed when the person is in the drowsy state. This will be notify according to the calculation of the specified value.

Drowsiness has been identified and shown in the Figure 5.3 which illustrates it graphically. Whereas the orange wave on the line graph represents the ratio value for the Mouth over Eye aspect ratio and the blue color wave represents the pupil circularity ratio. The final value of mouth over eye is distinctive in detection because the ratio value of eye is inversely propositional to the ratio value of mouth.

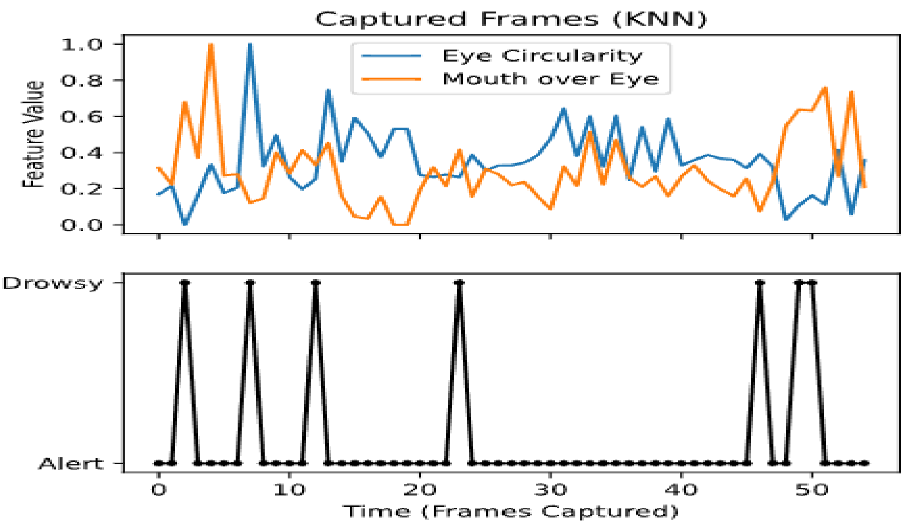


Figure 5.3 Graphical Representation of MOE, PUC

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

CONCLUSION

Here by, We have proposed by calculating eye ratio value, ratio value of mouth combines gives the mouth ratio over the eye ratio using the certain landmarks which are marked in the face. And Pupil Circularity is calculated in beyond Eye Aspect Ratio. Then the values are classified through the algorithm of KNN classifier. It is used in prediction of high range of accuracy. By using KNN we predicted the accuracy in range of 90.74%. The state of drowsiness is detected effectively.

Throughout this project we had learned quite a few things. First, simpler models can be just as efficient at completing tasks as more complex models. In our case, the K-Nearest Neighbor model gave an accuracy similar to the certain model. However, because we do not want to misclassify people who are drowsy as alert, ultimately it is better to use the more complex model with a lower false-negative rate than a simpler model that may be cheaper to deploy. Second, normalization was crucial to our performance.

Recognized that everybody has a different baseline for eye and mouth aspect ratios and normalizing for each participant was necessary. Outside of runtime for our models, data pre-processing and feature extraction/normalization took up a bulk of our time. It will be interesting to update our project and look into how we can decrease the false-negative rate for KNN and other simpler models.

FUTURE WORK

For future reference there are different level of investigations can be done. Boundary level of drowsiness are quite narrow, so transferring learning technique can be used to improve the performance. Also, the detection can be done by calculating the head pose, chin pose of a person. In these cases, it has a chance of getting higher accuracy. They are several types of supervised algorithms are there it can also implemented for future work to get better result.

Moving forward, there are a few things that we can do for further to improve our results and fine-tune the models. First, need to incorporate distance between the facial landmarks to account for any movement by the subject in the video. Realistically the participants will not be static on the screen and we believe sudden movements by the participant may signal drowsiness or waking up from micro-sleep. Second, we want to update parameters with our more complex models ensembles in order to achieve better results. Third and finally, we would like to collect own training data from a larger sample of participants while including new distinct signals of drowsiness like sudden head movement, hand movement, or even tracking eye movements.

APPENDIX

SAMPLE CODE

Functions

```
def eye_aspect_ratio(eye):
    """Calculates EAR -> ratio length and width of eyes"""
    A = distance.euclidean(eye[1], eye[5])
    B = distance.euclidean(eye[2], eye[4])
    C = distance.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear

def mouth_aspect_ratio(mouth):
    """Calculates MAR -> ratio length and width of mouth"""
    A = distance.euclidean(mouth[14], mouth[18])
    C = distance.euclidean(mouth[12], mouth[16])
    mar = (A ) / (C)
    return mar

def circularity(eye):
    """Calculates PUC -> low perimeter leads to lower pupil"""
    A = distance.euclidean(eye[1], eye[4])
    radius  = A/2.0
    Area = math.pi * (radius ** 2)
    p = 0
    p += distance.euclidean(eye[0], eye[1])
    p += distance.euclidean(eye[1], eye[2])
    p += distance.euclidean(eye[2], eye[3])
    p += distance.euclidean(eye[3], eye[4])
    p += distance.euclidean(eye[4], eye[5])
    p += distance.euclidean(eye[5], eye[0])
    return 4 * math.pi * Area / (p**2)
```

```
def mouth_over_eye(eye):
    """Calculates the MOE -> ratio of MAR to EAR"""
    ear = eye_aspect_ratio(eye)
    mar = mouth_aspect_ratio(eye)
    mouth_eye = mar/ear
    return mouth_eye

def calibration(detector, predictor, cap = cv2.VideoCapture(0)):
    """Helper function for determing mean and std"""

    font                   = cv2.FONT_HERSHEY_SIMPLEX
    bottomLeftCornerOfText = (10,400)
    fontScale              = 1
    fontColor              = (255,255,255)
    lineType               = 2
    data = []
    cap = cap

    while True:
        # Getting out image by webcam
        _, image = cap.read()
        # Converting the image to gray scale
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

        # Get faces into webcam's image
        rects = detector(image, 0)

        # For each detected face, find the landmark.
        for (i, rect) in enumerate(rects):
            # Make the prediction and transfom it to numpy array
            shape = predictor(gray, rect)
            shape = face_utils.shape_to_np(shape)
            data.append(shape)
            cv2.putText(image,"Calibrating...", bottomLeftCornerOfText, font, fontScale,
fontColor,lineType)

            # Draw on our image, all the finded cordinate points (x,y)
            for (x, y) in shape:
                cv2.circle(image, (x, y), 2, (0, 255, 0), -1)
        # Show the image
        cv2.imshow("Output", image)

        if cv2.waitKey(1)==ord('q'):
            break

    cv2.destroyAllWindows()
    cap.release()

    features_test = []
    for d in data:
        eye = d[36:68]
        ear = eye_aspect_ratio(eye)
        mar = mouth_aspect_ratio(eye)
        cir = circularity(eye)
        mouth_eye = mouth_over_eye(eye)
        features_test.append([ear, mar, cir, mouth_eye])

    features_test = np.array(features_test)
    x = features_test
    y = pd.DataFrame(x, columns=["EAR","MAR","Circularity","MOE"])
    df_means = y.mean(axis=0)
    df_std = y.std(axis=0)

    return df_means, df_std
```

Standardization

```
import pandas as pd

class Standardization:

    def __init__(self, df_total):
        self.df_total = df_total

    #Functions for getting mean and std of each feature
    def calculate_Standardization(self):
        def mean_EAR(respondent):
            return df_means.loc[respondent]["EAR"]

        def mean_MAR(respondent):
```

```
    return df_means.loc[respondent]["MAR"]

def mean_Circularity(respondent):
    return df_means.loc[respondent]["Circularity"]

def mean_MOE(respondent):
    return df_means.loc[respondent]["MOE"]

def std_EAR(respondent):
    return df_std.loc[respondent]["EAR"]

def std_MAR(respondent):
    return df_std.loc[respondent]["MAR"]

def std_Circularity(respondent):
    return df_std.loc[respondent]["Circularity"]

def std_MOE(respondent):
    return df_std.loc[respondent]["MOE"]

#Separating the rows which are "Alert" only
df_alert = self.df_total[self.df_total["Y"] == 0]

#Creating separate dataframes for each participants's first, second and third "Alert" frame
df_alert_1 = df_alert.iloc[0::240, :]
df_alert_2 = df_alert.iloc[1::240, :]
df_alert_3 = df_alert.iloc[2::240, :]

#Merging them into one dataframe
alert_first3 = [df_alert_1,df_alert_2,df_alert_3]
df_alert_first3 = pd.concat(alert_first3)
df_alert_first3 = df_alert_first3.sort_index()

#Based on the first 3 "Alert" frames, calculating per participant the mean and std for each feature
pd.options.mode.chained_assignment = None
df_means = df_alert_first3.groupby("Participant")[["EAR", "MAR", "Circularity", "MOE"]].mean()
df_std = df_alert_first3.groupby("Participant")[["EAR", "MAR", "Circularity", "MOE"]].std()
```

```
        #Adding respondent-wise mean and std for each feature to each row in the original
dataframe
        self.df_total["EAR_mean"] = self.df_total["Participant"].apply(mean_EAR)
        self.df_total["MAR_mean"] = self.df_total["Participant"].apply(mean_MAR)
        self.df_total["Circularity_mean"] = self.df_total["Participant"].apply(mean_Circularity)
        self.df_total["MOE_mean"] = self.df_total["Participant"].apply(mean_MOE)

        self.df_total["EAR_std"] = self.df_total["Participant"].apply(std_EAR)
        self.df_total["MAR_std"] = self.df_total["Participant"].apply(std_MAR)
        self.df_total["Circularity_std"] = self.df_total["Participant"].apply(std_Circularity)
        self.df_total["MOE_std"] = self.df_total["Participant"].apply(std_MOE)
        self.df_total.head()
        print(self.df_total.shape)

        #Calculating now normalized features for each row in the original dataframe
        self.df_total["EAR_N"] = (self.df_total["EAR"] - self.df_total["EAR_mean"]) /
self.df_total["EAR_std"]
        self.df_total["MAR_N"] = (self.df_total["MAR"] - self.df_total["MAR_mean"]) /
self.df_total["MAR_std"]
        self.df_total["Circularity_N"] = (self.df_total["Circularity"] -
self.df_total["Circularity_mean"]) / self.df_total["Circularity_std"]
        self.df_total["MOE_N"] = (self.df_total["MOE"] - self.df_total["MOE_mean"]) /
self.df_total["MOE_std"]
        return self.df_total
```

Program execution

```
from drowsiness_imports import *
from drowsiness_functions import *
# Read in dlibs shape predictor for detecting facial landmarks
p = r"shape_predictor_68_face_landmarks.dat"
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(p)

def datacreation(number, label, detector, predictor, cap = cv2.VideoCapture(0)):

    font                   = cv2.FONT_HERSHEY_SIMPLEX
    bottomLeftCornerOfText = (10,400)
    fontScale              = 1
    fontColor              = (255,255,255)
```

```
lineType               = 2
data = []
cap = cap

while True:
    # Getting out image by webcam
    _, image = cap.read()
    # Converting the image to gray scale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Get faces into webcam's image
    rects = detector(image, 0)

    # For each detected face, find the landmark.
    for (i, rect) in enumerate(rects):
        # Make the prediction and transfom it to numpy array
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        data.append(shape)

        # Draw on our image, all the finded cordinate points (x,y)
        for (x, y) in shape:
            cv2.circle(image, (x, y), 2, (0, 255, 0), -1)

    # Show the image
    cv2.imshow("Output", image)

    if cv2.waitKey(1)==ord('q'):
        break

cv2.destroyAllWindows()
cap.release()
features_test = []
for d in data:
    eye = d[36:68]
    ear = eye_aspect_ratio(eye)
    mar = mouth_aspect_ratio(eye)
    cir = circularity(eye)
    mouth_eye = mouth_over_eye(eye)
    features_test.append([number ,ear, mar, cir, mouth_eye,label])

features_test = np.array(features_test)
```

```
    x = features_test
    y = pd.DataFrame(x)

    return y
label=0.0
pre=pd.DataFrame()
for number in range(1,4):
    print("This is Participant State :",label)
    cur= datacreation(number, label, detector, predictor, cv2.VideoCapture(0))
    pre=pd.concat([pre,cur])
    label = label + 5
pre.to_csv('data.csv',index=False)

df=pd.read_csv(r'data.csv',names=["Participant", "EAR", "MAR", "Circularity", "MOE",
"Y"])
df

#Reading the CSV back into a dataframe
df_total = pd.read_csv(r'data_drowsiness\merged\totalwithrespondent.csv')

#Reordering the columns
cols = df_total.columns.tolist()
cols = cols[-1:] + cols[4:5] + cols[:4]
df_total = df_total[cols]
df_total

# Check if a candidate has only 5.0 labeled data
t = df_total.drop_duplicates("Participant")
df_total = df_total[~df_total.Participant.isin(t[t.Y > 0].Participant)]
df_total.shape

from drowsiness_standardisation import Standardization
pd.options.mode.chained_assignment = None

# Standardization
df_total = Standardization(df_total).calculate_Standardization()
df_total.head()

# Saving the file to a CSV with all the information
df_total.to_csv(r'data_drowsiness\prepared\totalwithallinfo.csv',index=False)
```

```
# Saving the file to a CSV with all the information
df_main = df_total.drop(["EAR_mean","MAR_mean", "Circularity_mean", "MOE_mean",
"EAR_std", "MAR_std", "Circularity_std", "MOE_std"], axis=1)
df_main.to_csv(r'data_drowsiness\prepared\totalwithmaininfo.csv',index=False)

# read in prepared data
df = pd.read_csv(r'data_drowsiness\prepared\totalwithmaininfo.csv',sep=',')
participants = set(df.Participant)
df = df.drop(["Participant"], axis=1)
df = df[df.Y != 5.0]
df.loc[df.Y == 0.0, "Y"] = int(0)
df.loc[df.Y == 10.0, "Y"] = int(1)

train_percentage = 18/len(participants)
train_samples = int(len(df) * train_percentage)
test_samples = len(df) - train_samples

df_train = df[:train_samples]
df_test = df[-test_samples:]

X_test = df_test.drop(["Y"], axis=1)
y_test = df_test["Y"]

X_train = df_train.drop('Y', axis=1)
y_train = df_train['Y']
print(f'X_test: {X_test.shape} \ny_test: {y_test.shape} \nX_train: {X_train.shape} \ny_train:
{y_train.shape}')
df_train

# class distribution
label = 'Alert', 'Drowsy'
plt.figure(figsize = (8,8))
plt.subplot(121)
plt.title("Training Set")
plt.pie(df_train.groupby('Y').size(), labels = label, autopct='%1.1f%%', startangle=45,
colors={"grey", "darkgrey"})
plt.subplot(122)
plt.title("Test Set")
plt.pie(df_test.groupby('Y').size(), labels = label, autopct='%1.1f%%', startangle=90,
colors={"grey", "darkgrey"})
plt.show()
```

KNN classifier

```
def average(y_pred):
    """Averaging sequential frames for classifier"""
    for i in range(1, len(y_pred)-1):
        if i % 240 == 0 or (i+1) % 240 == 0:
            pass
        else:
            average = float(y_pred[i-1] + y_pred[i] + y_pred[i+1])/3
            if average >= 0.5:
                y_pred[i] = 1
            else:
                y_pred[i] = 0
    return y_pred

acc3_list = []
f1_score3_list = []
roc_3_list = []

# take 45 runs and save best one
for i in range(1, 45):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train, y_train)
    pred_KN = neigh.predict(X_test)
    pred_KN = average(pred_KN)
    y_score_3 = neigh.predict_proba(X_test)[:,1]
    acc3_list.append(accuracy_score(y_test, pred_KN))
    f1_score3_list.append(metrics.f1_score(y_test, pred_KN))
    roc_3_list.append(metrics.roc_auc_score(y_test, y_score_3))

neigh = KNeighborsClassifier(n_neighbors=acc3_list.index(max(acc3_list))+1)
print(f"Neighbors: {neigh.get_params()['n_neighbors']}")
neigh.fit(X_train, y_train)

def model_knn(landmarks):
    """Returns features and classification result"""
    features = pd.DataFrame(columns=["EAR","MAR","Circularity","MOE"])
    lan = landmarks[36:68] # Extracting relevant parts (eyes + mouth)
    ear = eye_aspect_ratio(lan)
    mar = mouth_aspect_ratio(lan)
    cir = circularity(lan)
    mouth_eye = mouth_over_eye(lan)
    df=features.append({"EAR":ear,"MAR":mar,"Circularity":cir,"MOE":
mouth_eye},ignore_index=True)

    # Normalisation
    df["EAR_N"] = (df["EAR"] - mean["EAR"]) / std["EAR"]
    df["MAR_N"] = (df["MAR"] - mean["MAR"]) / std["MAR"]
    df["Circularity_N"] = (df["Circularity"] - mean["Circularity"]) / std["Circularity"]
    df["MOE_N"] = (df["MOE"] - mean["MOE"]) / std["MOE"]

    Result = neigh.predict(df)
    if Result == 1:
        Result_String = "Drowsy"
        fontColor = (0, 0, 245)
    else:
        Result_String = "Alert"
        fontColor = (255, 255, 255)

    return Result_String, df.values, fontColor
```

Live demo

```
def live(cnn=False):
    cap=cv2.VideoCapture(0)
    """real-time drowsiness recognition"""
    font                   = cv2.FONT_HERSHEY_SIMPLEX
    fontScale              = 1
    bottomLeftCornerOfText = (10, 400)
    lineType               = 2
    cap                    = cap
    max_prediction         = -1
    frame_counter          = 0
    data                   = []
    result                 = []
    result_bin             = []

    while True:
        # Getting out image by webcam
        _, image = cap.read()
    if not cnn:
        # Applying metric based approach
        # Converting the image to gray scale
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        # Get faces into webcam's image
        rects = detector(image, 0)
         # For each detected face, find the landmark.
        for (i, rect) in enumerate(rects):
            # Make the prediction and transfom it to numpy array
            shape = predictor(gray, rect)
            shape = face_utils.shape_to_np(shape)
            result_string, features, fontColor = model_knn(shape)

            cv2.putText(image, result_string, bottomLeftCornerOfText, font, fontScale,
fontColor,lineType)
            data.append (features)
            result.append(result_string)

            # Draw on our image, all the finded cordinate points (x,y)
            for (x, y) in shape:
                cv2.circle(image, (x, y), 2, (0, 255, 0), -1) # maybe changing red if drowsy,
otherwise green
    else:
            # Applying image based approach
            prediction = prediction_cnn(image)
            if (len(result_bin) < 40):
                if (max_prediction == -1):
                    cv2.putText(image, "Calibration...", bottomLeftCornerOfText, font, fontScale,
(255, 255, 255), lineType)
                else:
                    cv2.putText(image, result_string, bottomLeftCornerOfText, font, fontScale,
fontColor, lineType)
            else:
                # Majority voting based CNN detection using FIFO
                max_prediction = np.argmax(np.bincount(result_bin))
                if (max_prediction == 1):
                    result_string = "Drowsy"
                    fontColor = (0, 0, 245)
                else:
                    result_string = "Alert"
                    fontColor = (235, 0, 0)
                cv2.putText(image, result_string, bottomLeftCornerOfText, font, fontScale,
fontColor, lineType) #optional
                result.extend([result_string for i in range(frame_counter)])
                frame_counter = 0
                for i in range(int(20/2)):
                    result_bin.pop(0)
            frame_counter += 1
            result_bin.append(prediction)

    # Show the image
    cv2.imshow("Output", image)

    if cv2.waitKey(1)==ord('q'):
        break

cv2.destroyAllWindows()
cap.release()

return data, result

# Run Calibration (only necessary for KNN classifier)
mean, std = calibration(detector, predictor, cv2.VideoCapture(0))
for i, j in zip(mean, std):
print("Mean {:4f} and standard deviation {:4f}".format(i, j))

features, result = live(cnn=False)
```

Results

```
features =np.vstack(features)
y=pd.DataFrame(features,columns=["EAR","MAR","Circularity","MOE","EAR_N","MAR_
N","Circularity_N","MOE_N"])
y = y.drop(columns=["EAR_N","MAR_N","Circularity_N","MOE_N"])

x = y.values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
y = pd.DataFrame(x_scaled, columns=["Eye Aspect Ratio","Mouth Aspect Ratio","Eye
Circularity","Mouth over Eye"])

y["Result"] = result
```

```
fig, (ax1, ax2) = plt.subplots(nrows=2,
                               ncols=1,
                               sharex=True,
                               sharey=False,
                               figsize=(5, 5))

ax1.set_title("Captured Frames (KNN)")
ax1.plot(y["Eye Circularity"])
ax1.plot(y["Mouth over Eye"])
ax1.legend(("Eye Circularity", "Mouth over Eye"), loc="best")
ax1.set_ylabel('Feature Value')

ax2.plot(y["Result"],marker = '.', color = "Black")
ax2.set_xlabel('Time (Frames Captured)')

pred_KN = neigh.predict(X_test)
pred_KN = average(pred_KN)
y_score_3 = neigh.predict_proba(X_test)[:,1]
acc3 = accuracy_score(y_test, pred_KN)
f1_score_3 = metrics.f1_score(y_test, pred_KN)
roc_3 = metrics.roc_auc_score(y_test, y_score_3)
print("Accuracy: {:2f} \nF1-Score: {:2f}".format(acc3*100, f1_score_3*100))
print("ROC: {:2f}".format(roc_3*100))
print("Confusion Matrix")
print(confusion_matrix(y_test, pred_KN))
print(f"Correct classified items {np.trace(confusion_matrix(y_test, pred_KN))} ")
print(f"Based on {y_test.shape[0]} test records")
```

REFERENCES

[1] Nitish Bhardwaj, Mohsen Babaeian, Bianca Esquivel, and Mohammad Mozumdar,(2019), 'Real time driver drowsiness detection using a Logistic Regression based machine learning algorithm'.

[2] C. M.Sheela Rani, B. Mohana, (2019), 'Drowsiness detection based on eye closure and yawning detection'

[3] Venkata Krishna Kishore Kolli, Venkata Rami Reddy Chirra, Sronivasulu Reddy Uyyala, (2019), 'Machine learning approach for driver drowsiness detection based on eye state'.

[4] Mohammed Shinoy, Mohammed Kharbeche, RatebJabber, (2020), 'Driver drowsiness detection model using convolutional neural networks techniques for an android application'.

[5] Fudail Hasan, Alexey ashevnik,(2021), 'State Of The Art Analysis Of Modern Drowiness Detection Algorithm Based On Computer Version'.

[6] Alireza Mirrashid, Maryam Hashemi, Aliasghar Beheshtin Shirazi (2021), 'Driver safety development real time driver drowsiness detection system based on convolutional neural network'.

[7] Abd Kadir, Saravanaraj, Sathasivam, Mahamad Sharifah Saon, (2020), 'Drowsiness detection system using Eye Aspect Ratio technique'.

[8] Sukrit Mehta, Sharad Dadhich, Sahil Gumber, Arpita Jadhav Bhatt, (2019), 'Real time driver drowsiness detection system using eye aspect and eye closure ratio Real Time Driver Drowsiness Detection System Using Eye Aspect Ratio and Eye Closure Ratio'.

[9] Surekha Reddy, Manjunath Kotari, (2019), 'Driver drowsiness detection using machine learning approach'.

[10] Joseph Felix, Novie Theresia Br. Pasaribu, Agus Prjono, Ratnadewi, Roy Pramono Adhie, (2019), ‘Drowsiness detection according to thr number of blinking eyes specified from eye aspect ratio value modification’

[11] João Mateus Marques Santana, Caio Bezerra Souto Maior Marcio jose das Chagas Moura, Isis Didier Lins,(2021), ‘Real time classification for autonomous drowsiness detection using eye aspect ratio’.

[12] Jun Wang, Jialing Feng, Zhexiao Guo and Guo Dan, (2020), ‘Using eye aspect ratio to enhance fast and objective assessment of facial paralysis’.