ABSTRACT

Face recognition has been a very active research area since 1960, with attempts to make a machine recognize faces in the same way that we do. Because it improves productivity and is reliable, it can be used in various applications for identification and security.

This project aims to create a product that requires minimal input data and accurately recognizes faces.

Face recognition is divided into two parts: face detection and face identification. Each face from the input is detected using RetinaFace and compared to the data stored using DeepFace, with the recognized faces being marked as present in an excel sheet.

Keywords:
Face recognition, face detection, RetinaFace, face identification, DeepFace

CONTENTS

Chapter 1
INTRODUCTION

In recent times, we have seen that artificial intelligence and machine learning have been at the pinnacle of computer design, learning, and hardware. It is linked with computer vision. As we know that humans are capable of seeing things and visualizing them, through machine learning we make the computer system efficient enough to recognize and visualize things just like human beings do, and that too in a more efficient way through various efficient machine learning algorithms. As we know in today's world, the physical method of calling attendance and then noting it down is a time-consuming process that requires some effort. Now that we are moving ahead rapidly towards greater technology
In this project we implement facial recognition to record attendance by detecting faces in the inputs and comparing them to previously collected data

1.1. Scope

A method for accurately and quickly recording attendance with minimal input data and the ability to recognise an individual's face. It can be used in schools, universities, and the workplace to save time and effort, resulting in higher productivity.

1.2. Impact

It reduces the time and effort required to keep physical records.
Facial Recognition authentication avoids the need for physical contact, which is required for other biometric authentication methods. This has become a source of concern in recent years, following the pandemic.
Finally, because all records are kept in digital format, no paper is used, saving the environment.

Chapter 2
PROBLEM DEFINITION

2.1. Problem

- The conventional method of tracking attendance is to manually enter it into records, which wastes time and results in inaccuracy.
- During the pandemic, the use of biometrics such as fingerprint authentication is a source of concern.

2.2. Solution

Face Recognition technology could be used to create an automatic attendance system that makes counting and recognising people easier, more convenient, and more accurate without requiring physical contact.

Chapter 3
LITERATURE REVIEW

3.1. Online Attendance Marking System Using Facial Recognition and Intranet Connectivity

Authors:

Vibhanshu Pant ,Deepak Sharma ,Annapurani K ,R.Sundar

Content:

In this Facial Detection System using OpenCv was used and the Face Recognition using Local Binary Pattern Histogram(LBPH) recognizer class was used for the face recognition part.

Inference:

It was inferred that Facial Recognition via OpenCv is inefficient and not the most accurate model. Local Binary Pattern Histogram(LBPH) has a very low accuracy of 43%.

3.2. Automatic Student Attendance System Using Face Recognition

Authors:

Partha Chakraborty, Chowdhury Shahriar Muzammel, Mahmuda Khatun,SK. Fahmida Islam,Saifur Rehman

Content:

In this Facial Detection System using OpenCv and HaarCascadeClassifier was used and for the Face Recognition using Facial Recognition using Principal Component Analysis(PCA) was used.

Inference:

It was inferred that OpenCv is inefficient and overall process involving Eigenvectors is complicated.The overall accuracy is 80.22%

3.3. A Face Recognition System for Attendance Record in a Nigerian University

Authors:

Emeka Ogbuju(Federal University,Lokoja)

Content:

In this , face detection is achieved through Histogram of Oriented Gradients and Local Binary Pattern Algorithm(LBPA) and Face Recognition is achieved through the Principal Component Analysis(PCA).The PCA Algorithm is then implemented into two databases which is AR database and FERET database.

Inference:

It was inferred that as AR and FERET come under data mining, it requires an in-depth knowledge of data mining.In this , a lot of data is required to recognize the detected faces through PCA and the main aim of our project is to get maximum accuracy through less training data.On the AR database the accuracy was 99.5% and on the FERET database it's accuracy was 85.15%.

Chapter 4
PROBLEM DEFINITION

Face recognition-based attendance systems use high-definition monitor video and other information technology to solve the problem of recognising actual faces for the purpose of taking attendance.

A simple web application that automatically records attendance using Face Recognition and exports the data to an Excel sheet. The two stages of the process are Face Detection and Face Recognition. Face detection is accomplished with the RetinaFace python library, which detects and aligns various types of faces. DeepFace is a Python library that uses only one image as training data and compares two images to verify identity.

Chapter 5
REQUIREMENTS

5.1. SOFTWARE REQUIREMENTS

- Python 3.9
- CV2 Python Library
- Pandas Python Library
- RetinaFace Python Library
- DeepFace Python Library

5.2. HARDWARE REQUIREMENTS

- Display 1280x720p
- Web-Cam 720p
- Dual Core Processor
- 4GB Memory (RAM)
- Nvidia MX 350 (recommended)

Chapter 6
METHODOLOGY

RetinaFace for Face Detection:

RetinaFace is a deep learning-based cutting-edge facial detector for Python coming with facial landmarks. Its detection performance is amazing even in the crowd as shown in the following illustration.

RetinaFace is the face detection module of the insightface project. The original implementation is mainly based on mxnet. Then, its tensorflow-based re-implementation was published by Stanislas Bertrand. So, this repo is heavily inspired by the study of Stanislas Bertrand. Its source code is simplified and it is transformed to pip compatible but the main structure of the reference model and its pre-trained weights are the same. It is bound in the DeepFace library of python.

DeepFace for Face Recognition:

DeepFace is the most lightweight face recognition and facial attribute analysis library for Python. The open-sourced DeepFace library includes all leading-edge AI models for face recognition and automatically handles all procedures for facial recognition in the background.

If you run face recognition with DeepFace, you get access to a set of features:

- *Face Verification:*

  The task of face verification refers to comparing a face with another to verify if it is a match or not. Hence, face verification is commonly used to compare a candidate's face to another. This can be used to confirm that a physical face matches the one in an ID document.

- *Face Recognition:*

  The task refers to finding a face in an image database. Performing face recognition requires running face verification many times.

- *Facial Attribute Analysis:*

  The task of facial attribute analysis refers to describing the visual properties of face images. Accordingly, facial attributes analysis is used to extract attributes such as age, gender classification, emotion analysis, or race/ethnicity prediction.

- *Real-Time Face Analysis:*

  This feature includes testing face recognition and facial attribute analysis with the real-time video feed of your webcam.

Chapter 7
EXPERIMENTATION AND TESTING

Face detection and face identification are the two parts of the facial recognition process.

7.1. Version 1

Face detection was done with OpenCV's Haar Cascade classifier, and face identification was done with the LBPH training model, but the accuracy of LBPH for face identification is very low.
According to literature review 3.1, LBPH is 43 percent accurate.
As a result, we used the scikit-learn python library to explore other training models.

7.2. Version 2

Face detection was done with OpenCV's Haar Cascade classifier, and face identification was done with scikit-learn’s SVC, Perceptron and
K-neighbors classifier learning models

Testing accuracy of each model

SVC is 50 percent accurate

```
model.fit(X_training, y_training)
predictions = model.predict(X_testing)

correct = 0
incorrect = 0
total = 0
for actual, predicted in zip(y_testing, predictions):
    total += 1
    if actual == predicted:
        correct += 1
    else:
        incorrect += 1

# Print results
print(f"Results for model {type(model).__name__}")
print(f"Correct: {correct}")
print(f"Incorrect: {incorrect}")
print(f"Accuracy: {100 * correct / total:.2f}%")
```

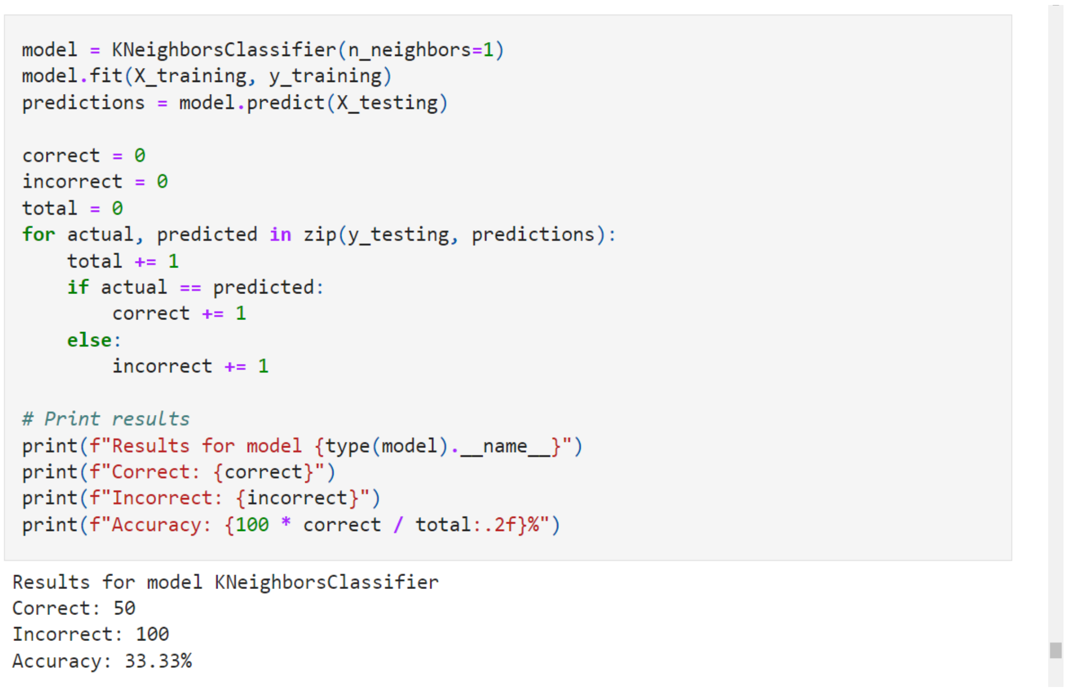
```
Results for model SVC
Correct: 75
Incorrect: 75
Accuracy: 50.00%
```

Perceptron is 66.67 percent accurate

```
model = Perceptron()
model.fit(X_training, y_training)
predictions = model.predict(X_testing)

correct = 0
incorrect = 0
total = 0
for actual, predicted in zip(y_testing, predictions):
    total += 1
    if actual == predicted:
        correct += 1
    else:
        incorrect += 1

# Print results
print(f"Results for model {type(model).__name__}")
print(f"Correct: {correct}")
print(f"Incorrect: {incorrect}")
print(f"Accuracy: {100 * correct / total:.2f}%")
```

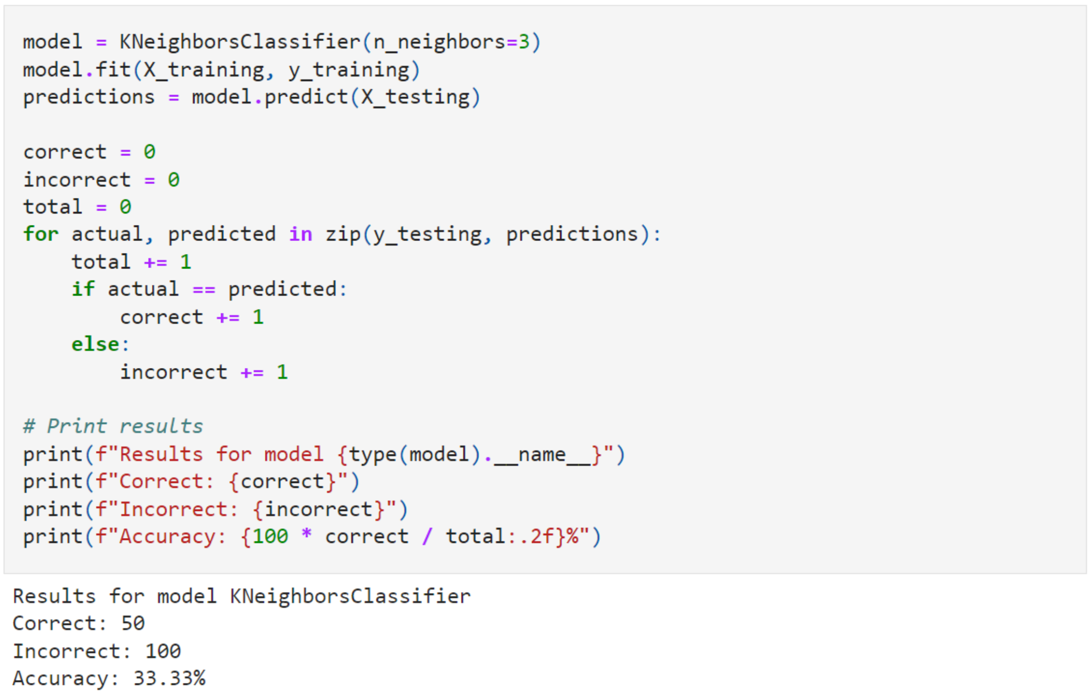
```
Results for model Perceptron
Correct: 100
Incorrect: 50
Accuracy: 66.67%
```

K-neighbors classifier is 33.33 percent accurate



```python
model = KNeighborsClassifier(n_neighbors=1)
model.fit(X_training, y_training)
predictions = model.predict(X_testing)

correct = 0
incorrect = 0
total = 0
for actual, predicted in zip(y_testing, predictions):
    total += 1
    if actual == predicted:
        correct += 1
    else:
        incorrect += 1

# Print results
print(f"Results for model {type(model).__name__}")
print(f"Correct: {correct}")
print(f"Incorrect: {incorrect}")
print(f"Accuracy: {100 * correct / total:.2f}%")
```

```
Results for model KNeighborsClassifier
Correct: 50
Incorrect: 100
Accuracy: 33.33%
```



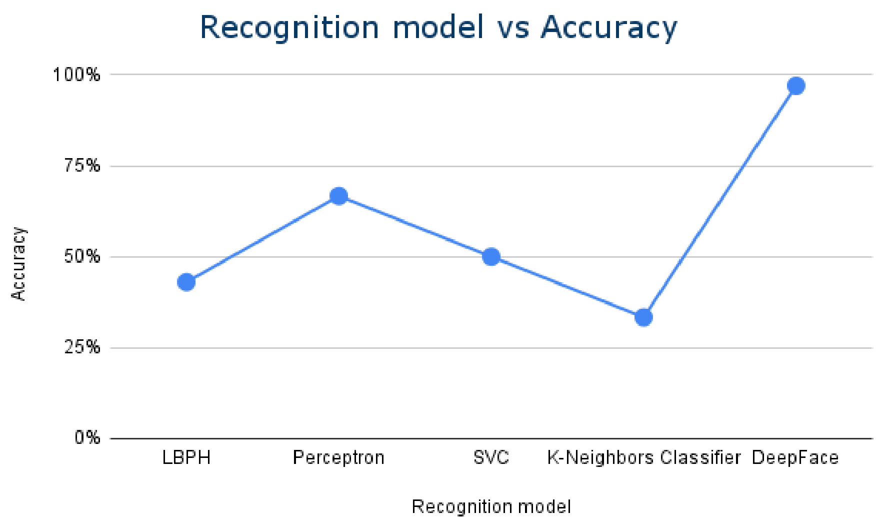
```python
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_training, y_training)
predictions = model.predict(X_testing)

correct = 0
incorrect = 0
total = 0
for actual, predicted in zip(y_testing, predictions):
    total += 1
    if actual == predicted:
        correct += 1
    else:
        incorrect += 1

# Print results
print(f"Results for model {type(model).__name__}")
print(f"Correct: {correct}")
print(f"Incorrect: {incorrect}")
print(f"Accuracy: {100 * correct / total:.2f}%")
```

```
Results for model KNeighborsClassifier
Correct: 50
Incorrect: 100
Accuracy: 33.33%
```

Despite the fact that SVC and Perceptron are more accurate than LBPH, they are still unreliable, so we implemented the DeepFace Python library, which claims to be 97.35 percent accurate.

When we compare the models, we get

Recognition model	Accuracy
LBPH	43%
Perceptron	66.67%
SVC	50%
K-Neighbors Classifier	33.33%
DeepFace	97%



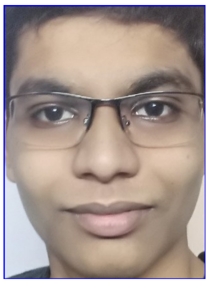
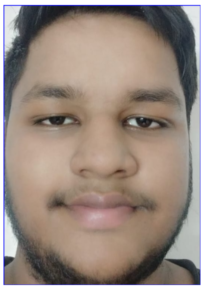
7.3. Version 3

Face detection is done with RetinaFace python library, which detects faces with 91.4 percent accuracy and also aligns them, improving identification accuracy. Face identification is done by DeepFace python library, which compares two images to verify identity, requiring only one input image and reducing the amount of stored data.

Chapter 8

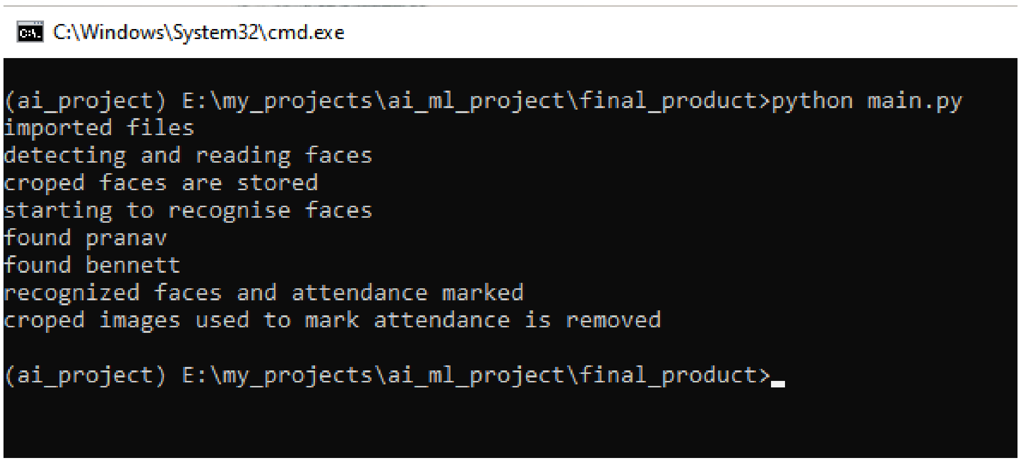
RESULT

Faces detected by RetinaFace

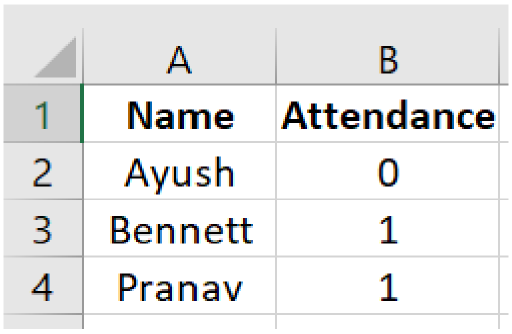



Output :

Result 1

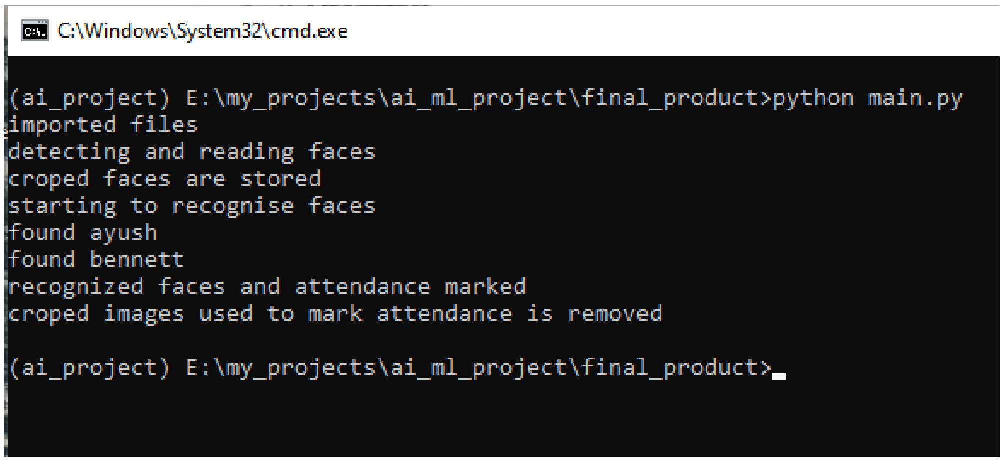


```
(ai_project) E:\my_projects\ai_ml_project\final_product>python main.py
imported files
detecting and reading faces
croped faces are stored
starting to recognise faces
found pranav
found bennett
recognized faces and attendance marked
croped images used to mark attendance is removed

(ai_project) E:\my_projects\ai_ml_project\final_product>_
```

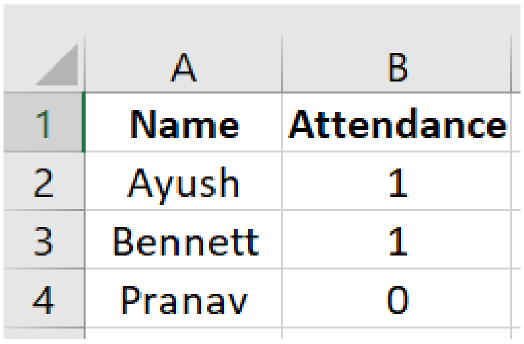


	A	B
1	Name	Attendance
2	Ayush	0
3	Bennett	1
4	Pranav	1

Result 2

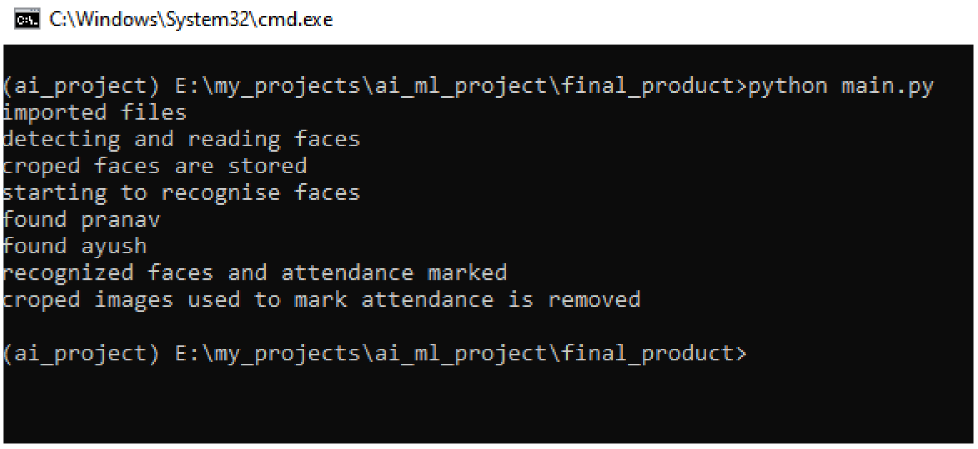


```
(ai_project) E:\my_projects\ai_ml_project\final_product>python main.py
imported files
detecting and reading faces
croped faces are stored
starting to recognise faces
found ayush
found bennett
recognized faces and attendance marked
croped images used to mark attendance is removed

(ai_project) E:\my_projects\ai_ml_project\final_product>
```

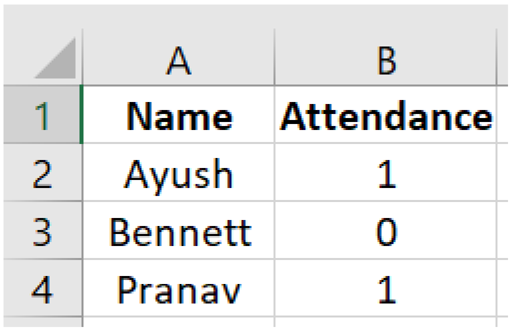


	A	B
1	Name	Attendance
2	Ayush	1
3	Bennett	1
4	Pranav	0

Result 3



```
(ai_project) E:\my_projects\ai_ml_project\final_product>python main.py
imported files
detecting and reading faces
croped faces are stored
starting to recognise faces
found pranav
found ayush
recognized faces and attendance marked
croped images used to mark attendance is removed

(ai_project) E:\my_projects\ai_ml_project\final_product>
```



	A	B
1	Name	Attendance
2	Ayush	1
3	Bennett	0
4	Pranav	1

REFERENCES

1 - Vibhanshu Pant ,Deepak Sharma ,Annapurani K ,R.Sundar, “Online Attendance Marking System Using Facial Recognition and Intranet Connectivity” In 2021 The authors and IOS Press, pages:2-4,doi:10.3233/APC210247,2021.

2 - Emeka Ogbuju(Federal University,Lokoja) “A Face Recognition System for Attendance Record in a Nigerian University”, In Journal Of Scientific Research and Development, 19[02], pages:38-45, https://www.researchgete.net/publication/342246965 - June, 2020.

3 - Partha Chakraborty, Chowdhury Shahriar Muzammel, Mahmuda Khatun,SK. Fahmida Islam,Saifur Rehman, “Automatic Student Attendance System using Face Recognition”, In International Journal of Engineering and Advanced Technology(IJEAT),” ISSN: 2249-8958, Volume-9 Issue-3”, https://www.researchgate.net/publication/339129579 - February, 2020.

OTHER REFERENCES

- https://cs50.harvard.edu/ai/2020/ ***(CS50 AI course on supervised learning and Sci-kit Learn)***
- https://github.com/opencv/opencv ***(OpenCV)***
- https://github.com/deepinsight/insightface/tree/master/detection ***(InsightFace's RetinaFace)***
- https://github.com/serengil/deepface ***(DeepFace)***