**Annexure-I**

### Title of the work

**MoodSync: Realtime Emotion Monitoring with Deep Learning**

**A Training Report**

Submitted in partial fulfillment of the requirements for the award of degree of

### **Name of Degree**

### **Machine Learning Specialization**

### **Submitted to**

**LOVELY PROFESSIONAL UNIVERSITY**

**PHAGWARA, PUNJAB**



### **From 06/05/23 to 08/25/23**

### **SUBMITTED BY**

#### **Name of student: Raja Saha**

Close-up of a handwritten sign

Description automatically generated**Registration Number: 12209471**

#### **Signature of the student:**

**Annexure-II: Student Declaration**

### To whom so ever it may concern

I, Raja Saha (12209471) thereby declare that the work done by me on “MoodSync: Realtime Emotion Monitoring with Deep Learning” from JUN’23 to AUG’23, is a record of original work for the partial fulfillment of the requirements for the award of the degree, MCA.

Raja Saha (12209471)

Close-up of a handwritten sign

Description automatically generated

Internship Certificate

(As given by MOOC or Organization in original)

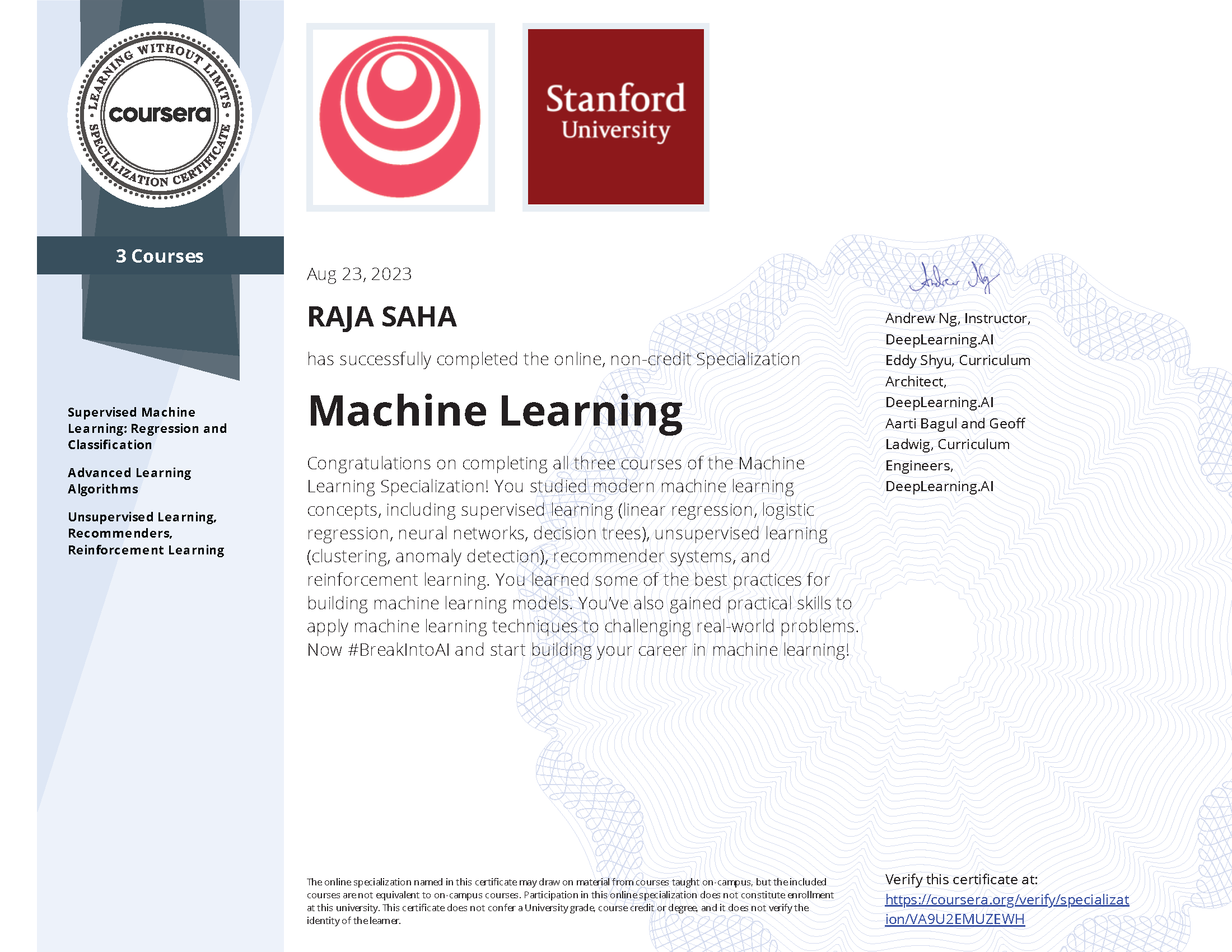


Table of Contents

[Title of the work 1](#_Toc143986722)

[**Name of Degree** 1](#_Toc143986723)

[**Machine Learning Specialization** 1](#_Toc143986724)

[**Submitted to** 1](#_Toc143986725)

[**From 06/05/23 to 08/25/23** 1](#_Toc143986726)

[**SUBMITTED BY** 1](#_Toc143986727)

[To whom so ever it may concern 2](#_Toc143986728)

[**INTRODUCTION OF THE COMPANY/WORK** 3](#_Toc143986729)

[**1.** **Introduction of the Course** 7](#_Toc143986730)

[**2.** **Technical Learnings from the course** 7](#_Toc143986731)

[**2.1** **Fundamental AI Concepts:** 7](#_Toc143986732)

[**2.2** **Deep Learning:** 7](#_Toc143986733)

[**2.3** **Machine Learning Algorithms:** 7](#_Toc143986734)

[**2.4** **Practical Implementation:** 8](#_Toc143986735)

[**2.5** **Certification:** 8](#_Toc143986736)

[**2.6** **Online Learning Platform** 8](#_Toc143986737)

[**2.7** **Supervised Learning:** 8](#_Toc143986738)

[**2.8** **Specialization Options:** 8](#_Toc143986739)

[**3** **Introduction of Mini Project** 8](#_Toc143986740)

[**4** **Details of Mini Project** 9](#_Toc143986741)

[**4.1** **The Need for MoodSync** 9](#_Toc143986742)

[**4.1.1** **Healthcare: Transforming Patient Care** 9](#_Toc143986743)

[**4.1.2** **Customer Service: Elevating Client Interactions** 10](#_Toc143986744)

[**4.1.3** **Education: Enriching Remote Learning** 10](#_Toc143986745)

[**4.1.4** **Entertainment: Personalizing Experiences** 10](#_Toc143986746)

[**4.2** **Technical Foundation** 10](#_Toc143986747)

[**4.2.1** **Dataset Utilization: The Fuel for Deep Learning** 11](#_Toc143986748)

[**4.2.2** **Deep Learning Model: The Cognitive Powerhouse** 11](#_Toc143986749)

[**4.2.3** **Real-time Emotion Detection: Swift and Precise** 11](#_Toc143986750)

[**4.2.4** **Labeling and User Interaction: Bridging the Communication Gap** 11](#_Toc143986751)

[**4.2.5** **High Accuracy: A Mark of Excellence** 11](#_Toc143986752)

[**5** **Interfaces Designed** 11](#_Toc143986753)

[**5.1** **Open Camera Interface** 12](#_Toc143986754)

[**5.1.1** **Objective:** 12](#_Toc143986755)

[**5.1.2** **Functionality:** 12](#_Toc143986756)

[**5.1.3** **User Experience:** 12](#_Toc143986757)

[**5.1.4** **Beta Phase Focus:** 12](#_Toc143986758)

[**5.2** **Real-time Emotion Detection** 13](#_Toc143986759)

[**5.2.1** **Objective:** 13](#_Toc143986760)

[**5.2.2** **Functionality:** 13](#_Toc143986761)

[**5.2.3** **User Experience:** 13](#_Toc143986762)

[**5.2.4** **Beta Phase Focus:** 13](#_Toc143986763)

[**5.3** **Examples:** 14](#_Toc143986764)

[**6** **Code snippets** 15](#_Toc143986765)

[**7** **Grade sheet of assignments/ marks card from the MOOC** 27](#_Toc143986766)

[**8** **Bibliography or References** 28](#_Toc143986767)

## **Introduction of the Course**

In an era where data drives innovation, the Machine Learning Specialization serves as a beacon of knowledge, guiding individuals from diverse backgrounds into the world of machine learning. This specialization is a curated blend of academic expertise and industry insights, making it a valuable asset for both beginners and experienced professionals.

The journey commences with the foundational course, "Machine Learning Foundations: A Case Study Approach." Here, learners gain a bird's-eye view of machine learning concepts, setting the stage for a deep dive into regression, classification, clustering, and more. Hands-on projects provide a solid foundation for practical applications.

The second course, "Machine Learning: Regression," propels learners into the intricacies of regression algorithms. Through engaging lectures and challenging assignments, participants sharpen their skills in predictive modeling—a vital skill in the data-driven world.

The final course, "Machine Learning: Classification," elevates learners to the status of machine learning practitioners. It delves into the fascinating world of classification, exploring algorithms that power recommendation systems, spam filters, and image recognition.

The distinctive feature of this specialization is the opportunity to work on real-world projects. These hands-on experiences enable learners to apply their knowledge to real data, solving practical problems and gaining the confidence to tackle complex machine learning challenges.

In conclusion, the Machine Learning Specialization from Stanford University is a transformative educational experience. It empowers learners with the knowledge and skills to harness the power of machine learning and contribute to the ever-evolving world of data science. Join this specialization and embark on a journey towards becoming a proficient machine learning practitioner.

## **Technical Learnings from the course**

* 1. **Fundamental AI Concepts:**

The Machine Learning Specialization (DeepLearning.AI) starts with fundamental AI concepts. This includes understanding what AI is, its history, and its significance in today's world. It lays the groundwork for more advanced topics in machine learning.

* 1. **Deep Learning:**

Deep learning is a critical component of the specialization. Deep neural networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs) are extensively covered. Learners gain insights into how deep learning powers applications like image recognition, natural language processing, and speech recognition.

* 1. **Machine Learning Algorithms:**

The course introduces learners to a wide range of machine learning algorithms. This includes decision trees, k-nearest neighbors, and clustering algorithms. Understanding these algorithms is essential for choosing the right approach for specific problems.

* 1. **Practical Implementation:**

The specialization emphasizes hands-on experience. Learners get the opportunity to implement machine learning models using tools like Python and libraries like TensorFlow and Keras. They work on real-world datasets, gaining practical skills that are directly applicable to projects.

* 1. **Certification:**

Completing the specialization awards learners with a certification from Stanford University and DeepLearning.AI. This certification is recognized and adds credibility to one's expertise in machine learning.

* 1. **Online Learning Platform**:

Coursera's user-friendly online platform ensures that learners have easy access to course materials. Video lectures, assignments, quizzes, and forums for discussions are readily available. The platform also allows for flexible learning schedules.

* 1. **Supervised Learning:**

The specialization includes a dedicated course on supervised learning. This deep dive into supervised learning includes topics like regression and classification, which are foundational to machine learning. It helps learners understand how machines can learn from labeled data.

* 1. **Specialization Options:**

Coursera offers a wide range of specializations and courses related to machine learning. Depending on one's interests and career goals, learners can choose from various options. For instance, the "Machine Learning for Trading Specialization" might be of interest to those in finance, while the "Machine Learning Specialization (UW)" offers a different perspective.

Overall, this specialization not only provides theoretical knowledge but also ensures that learners can practically apply what they've learned. The combination of fundamental concepts, deep learning, hands-on experience, and certification makes it a comprehensive resource for anyone looking to excel in machine learning. These technical learnings will be invaluable when working on your project report, providing a solid foundation for understanding and explaining machine learning concepts and techniques.

## **Introduction of Mini Project**

In an era characterized by the relentless march of technology, the ability to comprehend and react to human emotions is emerging as a pivotal factor across a multitude of sectors. Industries such as healthcare, customer service, education, and entertainment are increasingly recognizing the significance of understanding and responding to human emotions. It is within this context that MoodSync, our real-time emotion monitoring system, assumes its role as a cutting-edge solution.

MoodSync represents a confluence of advanced technologies, primarily deep learning and computer vision, all harnessed through the Python programming language. Its core functionality revolves around the real-time detection of human emotions through camera input, making it a potent tool in the realm of emotion analysis.

The need for such technology has never been more apparent. In healthcare, MoodSync can be a valuable asset, continuously monitoring the emotional states of patients and promptly identifying signs of distress or depression. In customer service, it can revolutionize the way companies interact with their clients by providing insights into customer satisfaction levels based on their facial expressions. In education, MoodSync empowers educators to gauge student engagement and emotional states during virtual learning sessions, offering immediate feedback on whether students are attentive or disinterested. In entertainment, this technology opens new frontiers by enabling content to adapt dynamically to the viewer's emotional state, offering personalized and immersive experiences.

At its core, MoodSync relies on a rich dataset called the "Face Expression Recognition Dataset" from Kaggle, a valuable resource for training deep learning models to recognize human emotions. This dataset, with its labeled facial expressions, forms the foundation for MoodSync's emotion recognition model. Deep learning and computer vision are the technical pillars that enable MoodSync to process and interpret live camera feeds in real-time, extracting emotional cues and providing valuable insights.

In summary, MoodSync is not just a mini project but a glimpse into the future of emotion analysis and real-time mood detection. Its technical sophistication, powered by deep learning and computer vision, positions it as a versatile tool with applications across various domains, addressing the increasingly critical need to understand and respond to human emotions in our digitally connected world.

## **Details of Mini Project**

## **The Need for MoodSync**

The compelling need for MoodSync arises from the rapidly evolving landscape of human-computer interaction. As technology continues to permeate every aspect of our daily lives, the capacity of machines to adeptly comprehend and respond to human emotional states becomes increasingly imperative. This necessity is particularly pronounced and pivotal across several domains.

### **Healthcare: Transforming Patient Care**

MoodSync emerges as a formidable asset within the realm of healthcare, where its applications are poised to revolutionize patient care. In healthcare settings, where patients' emotional well-being significantly influences their recovery, MoodSync takes on a critical role. By continuously monitoring the emotional states of patients, it performs the vital function of early emotional distress detection. For instance, if a patient within a hospital exhibits signs of profound sadness or anxiety, MoodSync can instantly alert healthcare providers, facilitating timely and precisely targeted interventions. This dynamic capability enhances the quality of care delivery, promoting not only the physical but also the emotional recovery of patients.

Moreover, in the burgeoning domain of telehealth, where remote consultations with therapists and medical professionals have become increasingly common, MoodSync offers an invaluable tool. During virtual counseling sessions, therapists can harness the real-time emotional feedback provided by MoodSync to gain insights into a patient's emotional well-being. This aids in the comprehensive assessment of a patient's mental health, enabling therapists to tailor their interventions with greater precision. MoodSync thus bridges the emotional gap created by virtual consultations, enhancing the effectiveness of remote therapy sessions.

### **Customer Service: Elevating Client Interactions**

In the arena of customer service, MoodSync heralds a transformative era by reshaping the dynamics of company-client interactions. Customer satisfaction is paramount in any business, and MoodSync introduces an innovative approach to ensure exceptional customer experiences. It accomplishes this through the astute analysis of customer facial expressions during interactions.

Consider a scenario where a customer engages with a customer service representative, but their facial expressions reveal frustration or dissatisfaction. MoodSync instantaneously detects these emotional cues and dispatches alerts to the customer service team. Armed with this real-time emotional feedback, customer service representatives can respond promptly and with precision, rectifying concerns and grievances before they escalate. This proactive approach not only enhances customer experiences but also fosters customer loyalty. Companies that employ MoodSync are better equipped to address customer concerns expeditiously, thus bolstering client retention and overall brand reputation.

### **Education: Enriching Remote Learning**

In the swiftly evolving landscape of education, particularly with the increasing prevalence of remote learning, MoodSync emerges as a potent tool. It empowers educators with the ability to gauge student engagement and emotional states during virtual learning sessions. This real-time emotional monitoring and feedback mechanism offer educators invaluable insights into their students' learning experiences.

Consider a virtual classroom scenario where a student's facial expressions indicate disinterest or frustration during an online class. MoodSync instantly provides feedback to the teacher, highlighting the emotional state of the student. Armed with this information, educators can pivot and adapt their teaching methods in real-time to re-engage and motivate the student effectively. MoodSync thus functions as an empathetic assistant to educators, ensuring that the remote learning environment remains dynamic, engaging, and conducive to effective learning outcomes.

### **Entertainment: Personalizing Experiences**

In the realm of entertainment, MoodSync heralds an era of personalization and immersion. It possesses the remarkable ability to dynamically adapt content based on the emotional state of the viewer, ushering in a new dimension of tailored experiences.

Imagine a scenario within a video game where a player's facial expressions reveal signs of fear or excitement. MoodSync, in response to these cues, can seamlessly adjust the game's difficulty level or alter the storyline to maximize the player's engagement and enjoyment. This real-time adaptation enhances the player's gaming experience by ensuring it remains challenging yet enjoyable.

In the realm of movie streaming services, MoodSync's capabilities extend to recommending films based not only on a viewer's past preferences but also on their current emotional state. If a viewer is displaying signs of melancholy, MoodSync can suggest heartwarming or uplifting films, ensuring that the viewer's emotional needs are met through their entertainment choices.

## **Technical Foundation**

At the heart of MoodSync's impressive capabilities lies a robust technical foundation grounded in deep learning and computer vision. These foundational technologies are adeptly harnessed to process and interpret live camera feeds in real-time, extracting emotional cues and providing invaluable insights.

### **Dataset Utilization: The Fuel for Deep Learning**

MoodSync initiates its journey by acquiring and meticulously preprocessing the "Face Expression Recognition Dataset" from Kaggle. This dataset stands as the bedrock upon which MoodSync's emotion recognition model is constructed. Comprising images of human faces, each meticulously labeled with one of seven different emotions—namely, angry, disgust, fear, happy, neutral, sad, and surprise—this dataset serves as the training and validation data for MoodSync's deep learning model.

### **Deep Learning Model: The Cognitive Powerhouse**

MoodSync harnesses the formidable capabilities of a Convolutional Neural Network (CNN), a neural network architecture celebrated for its prowess in image classification tasks. The CNN within MoodSync is a complex structure comprising multiple layers. It includes convolutional layers responsible for extracting intricate features from the facial images, as well as fully connected layers designed for the final classification of emotions. Through extensive training on the labeled dataset, the deep learning model within MoodSync evolves to recognize and understand the nuanced patterns and features associated with each of the seven emotions.

### **Real-time Emotion Detection: Swift and Precise**

Once fully trained, the deep learning model seamlessly integrates into the real-time emotion monitoring system at the heart of MoodSync. This system, characterized by its remarkable efficiency, can adeptly capture live camera feeds and process them on a frame-by-frame basis. Each frame undergoes intricate analysis, with MoodSync extracting facial features, detecting emotions, and promptly providing real-time feedback.

### **Labeling and User Interaction: Bridging the Communication Gap**

MoodSync possesses the capacity to translate the emotions it detects into readily understandable and human-readable labels. These labels, such as "happy" or "angry," serve as the medium through which the system communicates its insights to users. For instance, in a healthcare setting, MoodSync can alert healthcare providers with messages like "Patient appears distressed," thereby enabling swift and targeted interventions.

### **High Accuracy: A Mark of Excellence**

In conclusion, MoodSync is not merely a mini project but a visionary glimpse into the future of emotion analysis and real-time mood detection. Its technical sophistication, driven by the fusion of deep learning and computer vision, positions it as a versatile and transformative tool with applications across diverse domains. MoodSync rises to meet the increasingly critical need to comprehend and respond to human emotions in our digitally connected world.

Through its real-time insights into emotional states, MoodSync empowers healthcare providers to deliver personalized care, equips customer service representatives to elevate customer experiences, enables educators to optimize remote learning environments, and ushers in a new era of personalized entertainment. As the world continues its digital transformation, MoodSync stands at the forefront, harnessing these advancements to create a more emotionally intelligent and empathetic world.

### **Interfaces Designed**

In the current beta phase of our project, we have focused primarily on designing and implementing the fundamental interfaces for MoodSync. These interfaces serve as the gateway for users to access the core functionality of real-time emotion detection using a camera.

## **Open Camera Interface**

### **Objective:**

The primary objective of this interface is to seamlessly connect with the user's camera device, whether it's an integrated webcam or an external camera. It provides a live video feed from the camera to the system.

### **Functionality:**

* **Camera Access:** This interface establishes a connection with the camera device, ensuring that it is ready to capture video in real-time.
* **Live Video Feed:** Users are presented with a live video feed from the camera, allowing them to see themselves or the subject being monitored.
* **Real-time Processing:** As the video feed is displayed, MoodSync's core deep learning and computer vision algorithms are continuously processing each frame, extracting facial features, and detecting emotions in real-time.
* **User-Friendly:** The interface is designed to be user-friendly, requiring minimal interaction to initiate camera access. Users can simply click an "Open Camera" button to start the real-time emotion detection process.

### **User Experience:**

* **Seamless Transition:** Users experience a seamless transition from opening the interface to seeing themselves or the subject on the screen. There's no need for complicated setup or configurations.
* **Instant Feedback:** As soon as the camera feed starts, MoodSync begins providing real-time feedback about the detected emotional states, such as "happy," "angry," "sad," and more. This feedback is typically displayed in real-time alongside the video feed.

### **Beta Phase Focus:**

* **Stability and Performance:** In the beta phase, our primary focus for this interface is on ensuring its stability and performance. We aim to eliminate any potential bugs or glitches in the camera access and real-time processing components.
* **User Feedback:** We actively encourage beta testers to provide feedback on their experience with this interface. This input is invaluable in fine-tuning the interface for the final release.

## **Real-time Emotion Detection**

### **Objective:**

The central objective of MoodSync is to provide real-time emotion detection, and this is where the core processing takes place.

### **Functionality:**

* **Facial Analysis:** The interface performs intricate facial analysis on each frame from the camera feed, extracting facial features like the position of the eyes, eyebrows, mouth, and overall facial expressions.
* **Emotion Detection:** Utilizing the deep learning model, MoodSync accurately detects emotions based on the extracted facial features. It distinguishes between emotions such as happiness, sadness, anger, surprise, and more.
* **Feedback Display:** Detected emotions are promptly translated into human-readable labels and displayed to the user in real-time. For instance, if the system detects a happy expression, it will display "Happy" alongside the video feed.

### **User Experience:**

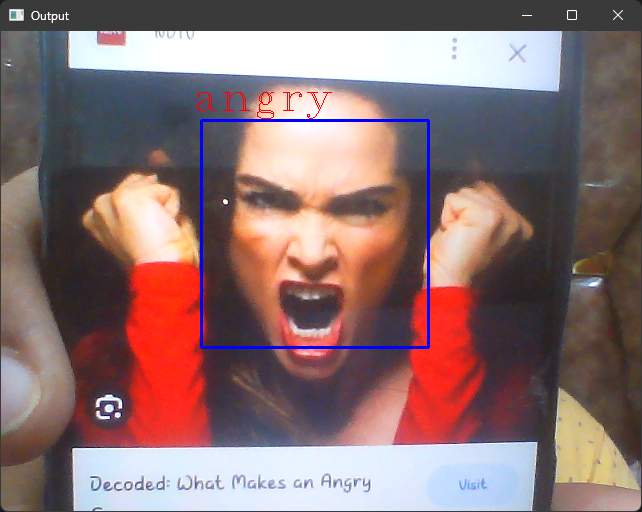
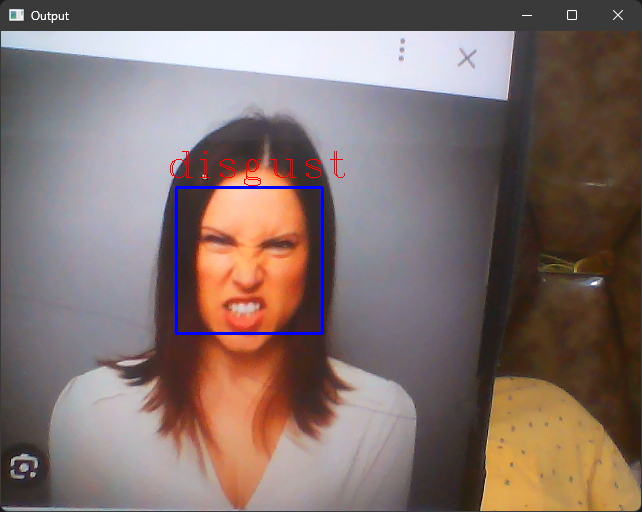
* **Immediate Insights:** Users receive immediate and actionable insights into the emotional states of themselves or the subject they are monitoring. This can be particularly valuable in various contexts, from personal well-being to professional interactions.

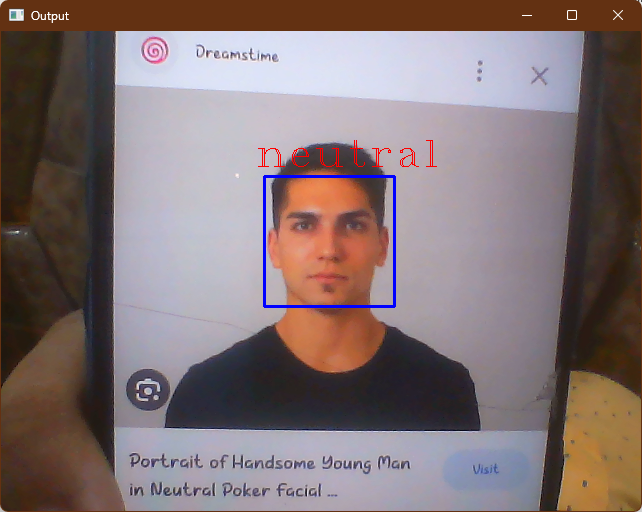
### **Beta Phase Focus:**

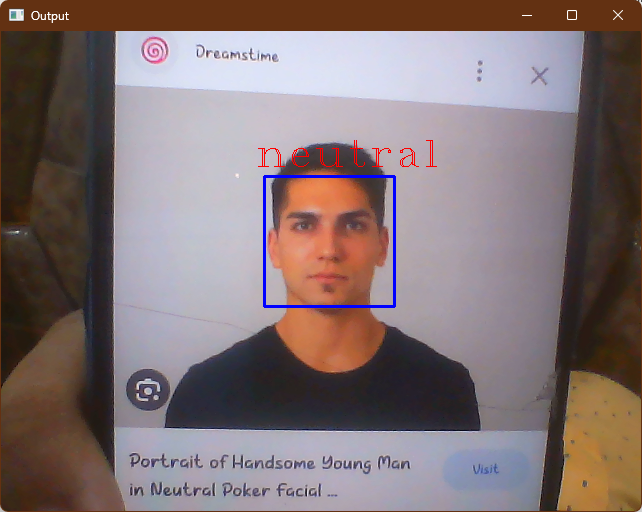
* **Accuracy:** In the beta phase, we are rigorously testing and refining the accuracy of emotion detection. We are continuously training and fine-tuning the deep learning model to improve its recognition capabilities.
* **Real-time Responsiveness:** We aim to ensure that the system's response to emotional cues is nearly instantaneous, enhancing the overall user experience.
* **Optimization:** Performance optimization is a key focus, ensuring that the real-time emotion detection runs smoothly and efficiently even on a variety of hardware configurations.

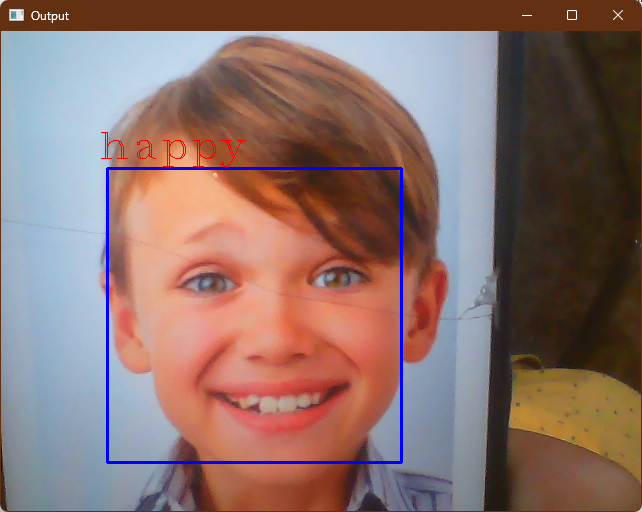
In summary, our beta phase interfaces, primarily focused on opening the camera and providing real-time emotion detection, are the foundational elements of MoodSync. These interfaces are designed to be accessible and user-friendly, delivering immediate insights into emotional states through live video feeds. As we gather user feedback and further refine these interfaces, MoodSync is poised to become a powerful tool for real-time mood monitoring and emotional analysis.

## **Examples:**









### **Code snippets**

File : “Emotion\_Detection.ipynb”

# %% [markdown]

# # MoodSync

# #### Realtime Emotion Monitoring with Deep Learning (With Picture Only)

# %% [markdown]

# ### Import necessary libraries for deep learning and data handling.

# %%

from keras.utils import to\_categorical  # Utilize one-hot encoding for target labels.

from tensorflow.keras.preprocessing.image import load\_img  # Load images for preprocessing.

from keras.models import Sequential  # Initialize a sequential model.

from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D  # Define layers for the neural network.

from PIL import Image  # Import the Python Imaging Library for image operations.

import os  # Access and manipulate the file system.

import pandas as pd  # Work with data in tabular form.

import numpy as np  # Perform numerical operations.

# %% [markdown]

# ### Define directory paths for training and testing data.

#

# %%

TRAIN\_dir='train'

TEST\_dir='test'

# %% [markdown]

# ### Define a function to create a DataFrame from image files in a directory.

# %%

*def* createDF(*dir*):

    image\_paths = []  # Store the paths of image files.

    labels = []  # Store corresponding labels.

    # Loop through each subdirectory (label) in the given directory.

    for label in os.listdir(dir):

        for imagename in os.listdir(os.path.join(dir, label)):

            image\_paths.append(os.path.join(dir, label, imagename))  # Store the image file path.

            labels.append(label)  # Store the label associated with the image.

        print(label, "completed")  # Print progress for each label.

    return image\_paths, labels

# %% [markdown]

# ### Create a DataFrame named 'train' to store image paths and labels.

# %%

train = pd.DataFrame()

# Populate the 'train' DataFrame using the 'createDF' function.

train['image'], train['label'] = createDF(TRAIN\_dir)

# %%

train

# %% [markdown]

# ### Create a DataFrame named 'test' to store image paths and labels.

#

# %%

test = pd.DataFrame()

# Populate the 'test' DataFrame using the 'createDF' function.

test['image'], test['label'] = createDF(TEST\_dir)

# %%

test

# %% [markdown]

# ### Import the 'tqdm' library to display progress bars in notebooks.

# %%

from tqdm.notebook import tqdm

# %% [markdown]

# ### Define a function to extract image features.

# %%

*def* extract\_features(*images*):

    features = []  # Initialize a list to store features.

    for image in tqdm(images):  # Iterate through the list of images with a progress bar.

        img = load\_img(image, *grayscale*=True)  # Load the image in grayscale.

        img = np.array(img)  # Convert the image to a NumPy array.

        features.append(img)  # Append the image to the features list.

    features = np.array(features)  # Convert the features list to a NumPy array.

    features = features.reshape(len(features), 48, 48, 1)  # Reshape the features array.

    return features  # Return the extracted features.

# %% [markdown]

# ### Extract features from the 'train' DataFrame's 'image' column.

# %%

train\_features = extract\_features(train['image'])

# %% [markdown]

# ### Extract features from the 'test' DataFrame's 'image' column.

# %%

test\_features = extract\_features(test['image'])

# %% [markdown]

# ### Normalize the pixel values of the training and test features.

# %%

x\_train = train\_features/255.0

x\_test = test\_features/255.0

# %% [markdown]

# ### Import the 'LabelEncoder' class from the 'sklearn.preprocessing' module.

# %%

from sklearn.preprocessing import LabelEncoder

# %% [markdown]

# ### Create a LabelEncoder instance and fit it to the labels in the 'train' DataFrame.

# %%

le = LabelEncoder()

le.fit(train['label'])

# %% [markdown]

# ### Transform the categorical labels in the 'train' and 'test' DataFrames to numerical values.

# %%

y\_train = le.transform(train['label'])

y\_test = le.transform(test['label'])

# %% [markdown]

# ### Convert the numerical labels to one-hot encoded format.

# %%

y\_train = to\_categorical(y\_train,*num\_classes* = 7)

y\_test = to\_categorical(y\_test,*num\_classes* = 7)

# %% [markdown]

# ### Create a Sequential model for deep learning.

# %%

# Create a Sequential model for deep learning.

model = Sequential()

# Add convolutional layers with max-pooling and dropout.

model.add(Conv2D(128, *kernel\_size*=(3, 3), *activation*='relu', *input\_shape*=(48, 48, 1)))

model.add(MaxPooling2D(*pool\_size*=(2, 2)))

model.add(Dropout(0.4))

model.add(Conv2D(256, *kernel\_size*=(3, 3), *activation*='relu'))

model.add(MaxPooling2D(*pool\_size*=(2, 2)))

model.add(Dropout(0.4))

model.add(Conv2D(512, *kernel\_size*=(3, 3), *activation*='relu'))

model.add(MaxPooling2D(*pool\_size*=(2, 2)))

model.add(Dropout(0.4))

model.add(Conv2D(512, *kernel\_size*=(3, 3), *activation*='relu'))

model.add(MaxPooling2D(*pool\_size*=(2, 2)))

model.add(Dropout(0.4))

# Flatten the output for fully connected layers.

model.add(Flatten())

# Add fully connected layers with dropout.

model.add(Dense(512, *activation*='relu'))

model.add(Dropout(0.4))

model.add(Dense(256, *activation*='relu'))

model.add(Dropout(0.3))

# Add the output layer with softmax activation for classification.

model.add(Dense(7, *activation*='softmax'))

# %% [markdown]

# ### Compile the neural network model.

# %%

model.compile(*optimizer* = 'adam', *loss* = 'categorical\_crossentropy', *metrics* = 'accuracy' )

# %% [markdown]

# ### Train the model on the training data.

# %%

model.fit(*x*= x\_train,*y* = y\_train, *batch\_size* = 128, *epochs* = 100, *validation\_data* = (x\_test,y\_test))

# %% [markdown]

# ### Serialize the model architecture to JSON and save it to a file.

# %%

model\_json = model.to\_json()

with open("emotiondetector.json", 'w') as json\_file:

    json\_file.write(model\_json)

# Save the model's weights and other parameters to an HDF5 file.

model.save("emotiondetector.h5")

# %% [markdown]

# ### Import the function to load a model from JSON in Keras.

# %%

from keras.models import model\_from\_json

# %% [markdown]

# ### Load the model architecture from a JSON file.

# %%

json\_file = open("facialemotionmodel.json", "r")

model\_json = json\_file.read()

json\_file.close()

# Create a new model from the loaded architecture.

model = model\_from\_json(model\_json)

# Load the model's weights from an HDF5 file.

model.load\_weights("facialemotionmodel.h5")

# %% [markdown]

# ### Define a list of emotion labels corresponding to the model's output classes.

# %%

label = ['angry','disgust','fear','happy','neutral','sad','surprise']

# %% [markdown]

# ### Define a function to extract features from a single image file.

#

# %%

*def* ef(*image*):

    img = load\_img(image, *grayscale*=True)  # Load the image in grayscale.

    feature = np.array(img)  # Convert the image to a NumPy array.

    feature = feature.reshape(1, 48, 48, 1)  # Reshape the feature.

    return feature / 255.0  # Normalize the pixel values.

# %%

# Define the path to the image file.

image = 'train/sad/42.jpg'

# Print the original emotion label.

print("Original image is of sad")

# Extract features from the image using the 'ef' function.

img = ef(image)

# Make a prediction using the trained model.

pred = model.predict(img)

# Get the predicted emotion label from the 'label' list.

pred\_label = label[pred.argmax()]

# Print the model's prediction.

print("Model prediction is", pred\_label)

# %% [markdown]

# ### Import the 'matplotlib' library for plotting and display in a Jupyter notebook.

# %%

import matplotlib.pyplot as plt

%matplotlib inline

# %%

# Define the path to the image file.

image = 'train/sad/19.jpg'

# Print the original emotion label.

print("Original image is of sad")

# Extract features from the image using the 'ef' function.

img = ef(image)

# Make a prediction using the trained model.

pred = model.predict(img)

# Get the predicted emotion label from the 'label' list.

pred\_label = label[pred.argmax()]

# Print the model's prediction.

print("Model prediction is", pred\_label)

# Display the image using matplotlib.

plt.imshow(img.reshape(48, 48), *cmap*='gray')

# %%

# Define the path to the image file.

image = 'train/fear/288.jpg'

# Print the original emotion label.

print("Original image is of fear")

# Extract features from the image using the 'ef' function.

img = ef(image)

# Make a prediction using the trained model.

pred = model.predict(img)

# Get the predicted emotion label from the 'label' list.

pred\_label = label[pred.argmax()]

# Print the model's prediction.

print("Model prediction is", pred\_label)

# Display the image using matplotlib.

plt.imshow(img.reshape(48, 48), *cmap*='gray')

# %%

# Define the path to the image file.

image = 'train/disgust/299.jpg'

# Print the original emotion label.

print("Original image is of disgust")

# Extract features from the image using the 'ef' function.

img = ef(image)

# Make a prediction using the trained model.

pred = model.predict(img)

# Get the predicted emotion label from the 'label' list.

pred\_label = label[pred.argmax()]

# Print the model's prediction.

print("Model prediction is", pred\_label)

# Display the image using matplotlib.

plt.imshow(img.reshape(48, 48), *cmap*='gray')

# %%

# Define the path to the image file.

image = 'train/happy/7.jpg'

# Print the original emotion label.

print("Original image is of happy")

# Extract features from the image using the 'ef' function.

img = ef(image)

# Make a prediction using the trained model.

pred = model.predict(img)

# Get the predicted emotion label from the 'label' list.

pred\_label = label[pred.argmax()]

# Print the model's prediction.

print("Model prediction is", pred\_label)

# Display the image using matplotlib.

plt.imshow(img.reshape(48, 48), *cmap*='gray')

# %%

# Define the path to the image file.

image = 'train/surprise/15.jpg'

# Print the original emotion label.

print("Original image is of surprise")

# Extract features from the image using the 'ef' function.

img = ef(image)

# Make a prediction using the trained model.

pred = model.predict(img)

# Get the predicted emotion label from the 'label' list.

pred\_label = label[pred.argmax()]

# Print the model's prediction.

print("Model prediction is", pred\_label)

# Display the image using matplotlib.

plt.imshow(img.reshape(48, 48), *cmap*='gray')

In the file "Emotion\_Detection.ipynb," we train a deep learning model using a pre-defined dataset to recognize human emotions from facial expressions. This notebook guides the model through a training process where it learns to identify emotions like happiness, sadness, and anger. After training, the notebook provides output demonstrating the model's ability to accurately classify emotions in real-world images.

File : Realtime Detection :

# %% [markdown]

# # MoodSync

# #### Realtime Emotion Monitoring with Deep Learning (With Camera)

# %%

# Import the 'cv2' library for computer vision operations.

import cv2

# Import the 'model\_from\_json' function from Keras for loading a model from JSON.

from keras.models import model\_from\_json

# Import the 'numpy' library for numerical operations.

import numpy as np

# %%

# Open the JSON file containing the model architecture for reading.

json\_file = open("facialemotionmodel.json", "r")

# Read the contents of the JSON file.

model\_json = json\_file.read()

# Close the JSON file.

json\_file.close()

# Create a new model from the loaded architecture.

model = model\_from\_json(model\_json)

# %%

# Load the model's weights from an HDF5 file.

model.load\_weights("facialemotionmodel.h5")

# Define the path to the Haar Cascade XML file for face detection.

haar\_file = cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml'

# Create a face cascade classifier using the Haar Cascade XML file.

face\_cascade = cv2.CascadeClassifier(haar\_file)

# %% [markdown]

# ### Define a function to extract features from an image.

# %%

*def* extract\_features(*image*):

    feature = np.array(image)  # Convert the image to a NumPy array.

    feature = feature.reshape(1, 48, 48, 1)  # Reshape the feature.

    return feature / 255.0  # Normalize the pixel values.

# %% [markdown]

# ### Open the webcam for video capture.

# %%

webcam = cv2.VideoCapture(0)

# Define a dictionary to map class indices to emotion labels.

labels = {0: 'angry', 1: 'disgust', 2: 'fear', 3: 'happy', 4: 'neutral', 5: 'sad', 6: 'surprise'}

# Main loop for real-time emotion recognition from webcam feed.

while True:

    # Read a frame from the webcam.

    i, im = webcam.read()

    # Convert the frame to grayscale for face detection.

    gray = cv2.cvtColor(im, cv2.COLOR\_BGR2GRAY)

    # Detect faces in the frame using Haar Cascade classifier.

    faces = face\_cascade.detectMultiScale(im, 1.3, 5)

    try:

        # Process each detected face.

        for (p, q, r, s) in faces:

            # Extract the face region and draw a rectangle around it.

            image = gray[q:q+s, p:p+r]

            cv2.rectangle(im, (p, q), (p+r, q+s), (255, 0, 0), 2)

            # Resize the face region to match the model's input size.

            image = cv2.resize(image, (48, 48))

            # Extract features from the resized image.

            img = extract\_features(image)

            # Use the trained model to predict the emotion label.

            pred = model.predict(img)

            prediction\_label = labels[pred.argmax()]

            # Overlay the predicted emotion label on the frame.

            cv2.putText(im, '% s' % (prediction\_label), (p-10, q-10), cv2.FONT\_HERSHEY\_COMPLEX\_SMALL, 2, (0, 0, 255))

        # Display the frame with face detection and emotion recognition results.

        cv2.imshow("Output", im)

        # Wait for a key press, exit the loop if 'Esc' is pressed.

        cv2.waitKey(27)

    except cv2.error:

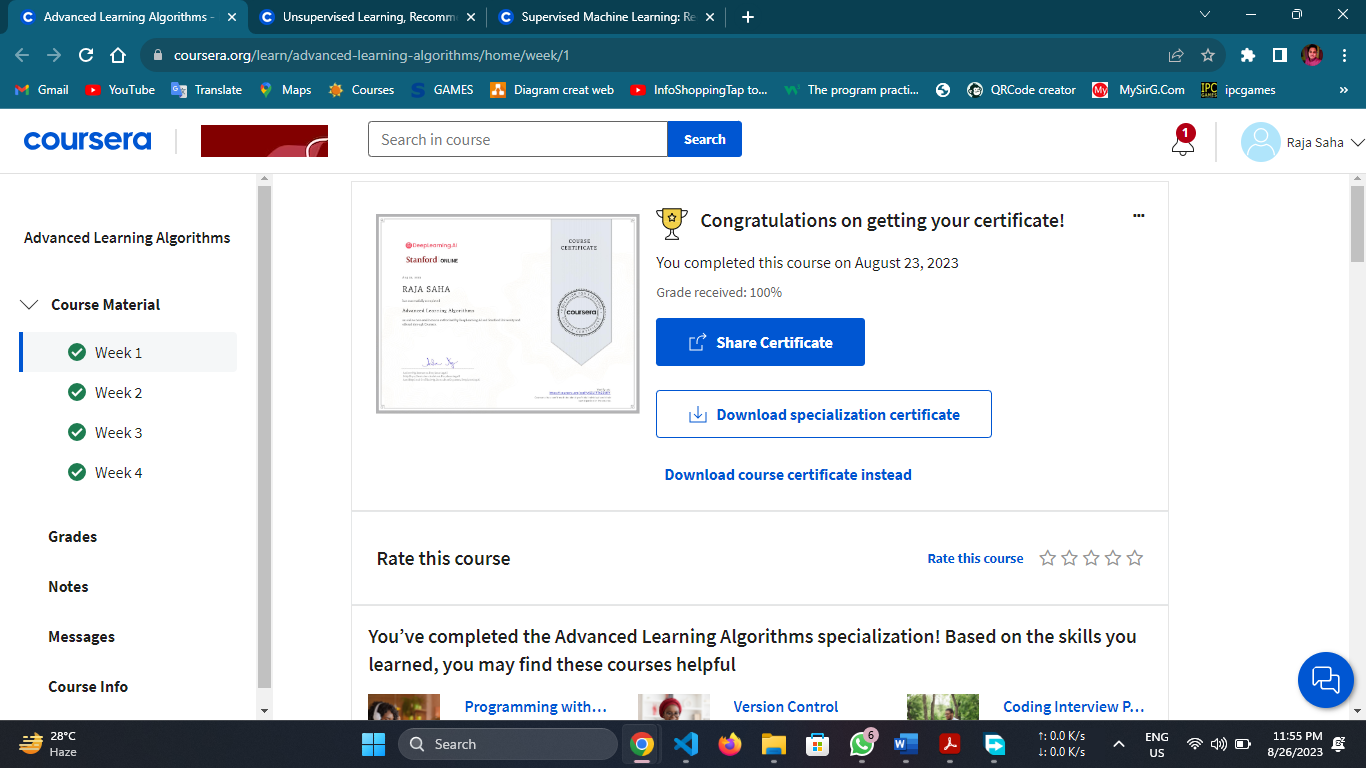
        # Handle errors (e.g., when no faces are detected).

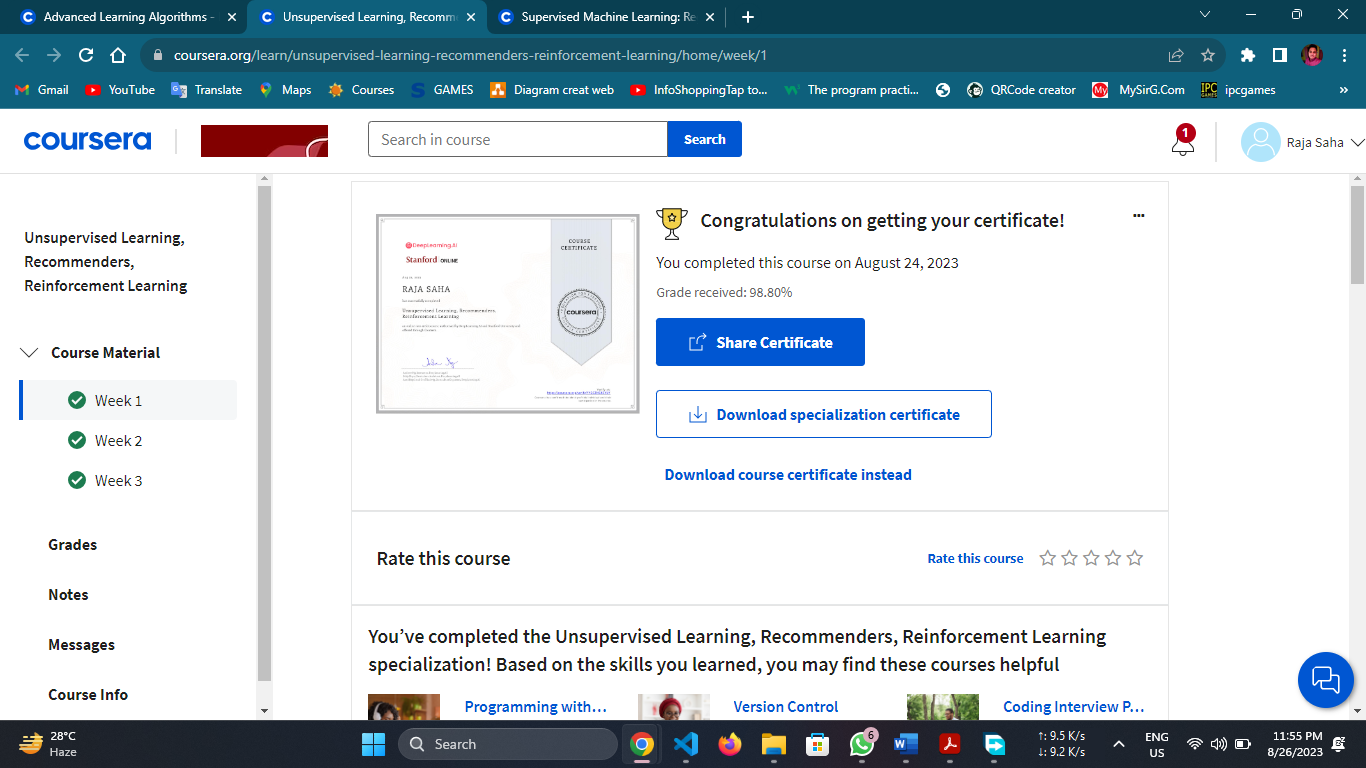
        pass

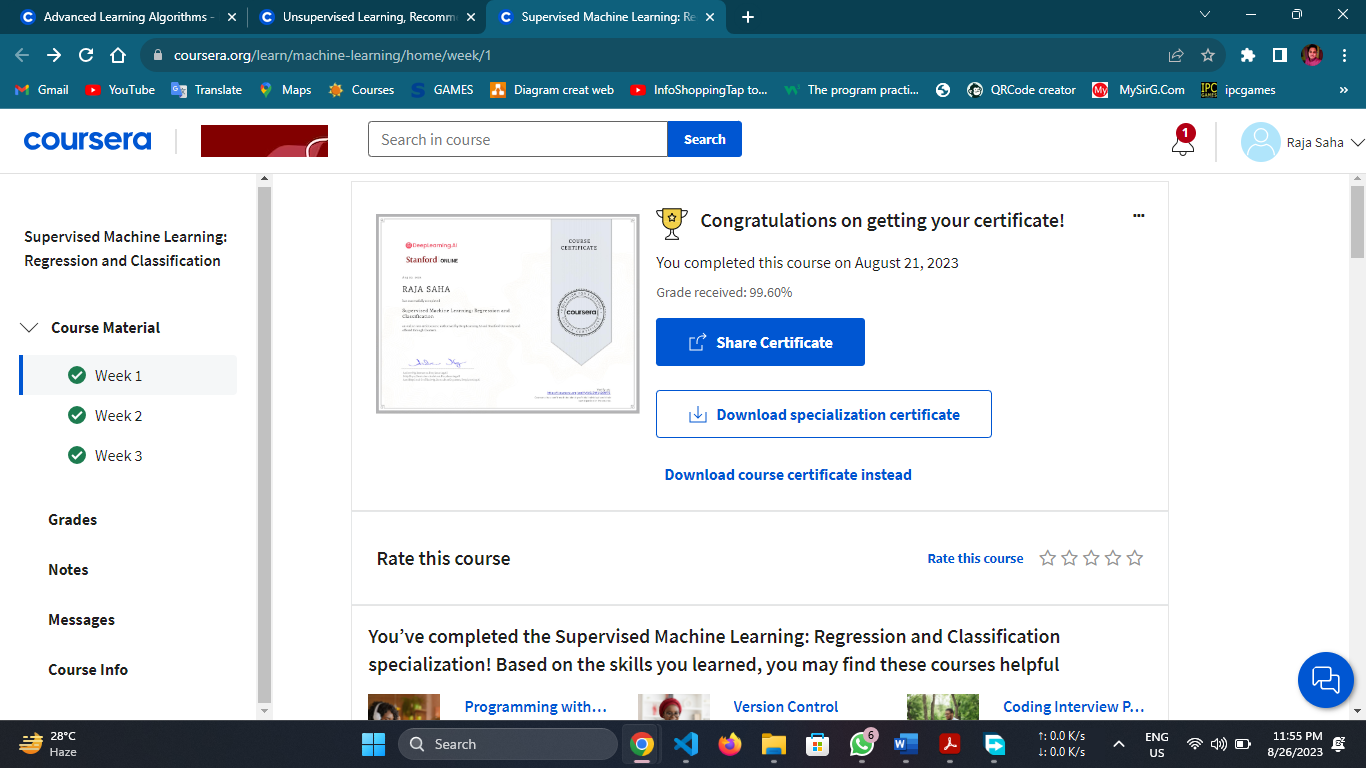
# %%

In the "RealtimeDetection.ipynb" file, our model undergoes a training process using predefined data to learn and improve its performance. Once trained, the model is applied to real-time input, such as live camera feeds, to generate predictions or outputs based on the patterns it has learned, enabling real-time emotion detection.

### **Grade sheet of assignments/ marks card from the MOOC**







### **Bibliography or References**

* Courcera