import ipywidgets as widgets

from PIL import Image

from IPython.display import display

# Creating widgets

text = widgets.Text(description='Enter image path:')

button = widgets.Button(description='Detect Emotion')

output = widgets.Output()

def preprocess\_image(image\_path):

    # Loading the image using PIL (Python Imaging Library)

    image = Image.open(image\_path)

    #Converting input image to grayscale as our model accepts only grayscale input

    image = image.convert('L')

    # Resizing the image to match the input size of your CNN model

    image = image.resize((48,48))

    # Converting the image to a NumPy array

    image\_array = np.array(image) / 255.0  # Normalize pixel values to [0, 1]

    # Adding an extra dimension to the array to represent the batch size=1

    image\_array = np.expand\_dims(image\_array, axis=0)

    return image\_array

def detect\_emotion(image\_path):

    # Preprocessing the input image

    image\_array = preprocess\_image(image\_path)

    # Using the CNN model to predict the emotion

    predicted\_emotion = np.argmax(model.predict(image\_array), axis=-1)

    # Mapping the predicted emotion index to the corresponding emotion label

    emotion\_labels = {0: 'Happy', 1: 'Neutral', 2: 'Sad', 3: 'Surprise'}

    #Mapping the predicted emotion to its label

    predicted\_emotion\_label = emotion\_labels[predicted\_emotion[0]]

    #returning the predicted\_emotion\_label

    return predicted\_emotion\_label

# Defining callback function for button click

def on\_button\_click(b):

    with output:

        # Performing emotion detection using the entered image path

        image\_path = text.value

        # Calling the detect\_emotion function to predict the emotion

        result=detect\_emotion(image\_path)

        # Printing the result

        print(f'Emotion detected in {image\_path}')

        print(result)

# Registering callback function with the button

button.on\_click(on\_button\_click)

# Displaying the widgets

display(text)

display(button)

display(output)