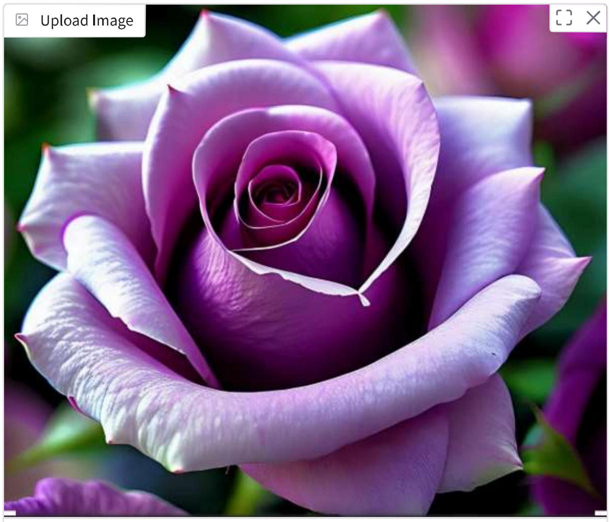
```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from PIL import Image
import gradio as gr

# -------- Built-in color dataset --------
data = {
    "color_name": [
        "Black", "White", "Red", "Lime", "Blue",
        "Yellow", "Cyan", "Magenta", "Silver", "Gray",
        "Maroon", "Olive", "Green", "Purple", "Teal", "Navy"
    ],
    "R": [0, 255, 255, 0, 0, 255, 0, 255, 192, 128, 128, 128, 0, 128, 0, 0],
    "G": [0, 255, 0, 255, 0, 255, 255, 0, 192, 128, 0, 128, 128, 0, 128, 0],
    "B": [0, 255, 0, 0, 255, 0, 255, 255, 192, 128, 0, 0, 0, 128, 128, 128],
}

labels_df = pd.DataFrame(data)

# -------- Match closest colors --------
def find_closest_colors(detected_colors, dataset):
    matches = []
    for color in detected_colors:
        r1, g1, b1 = color
        dataset_copy = dataset.copy()
        dataset_copy["distance"] = ((dataset_copy["R"] - r1) ** 2 +
                                    (dataset_copy["G"] - g1) ** 2 +
                                    (dataset_copy["B"] - b1) ** 2) ** 0.5
        closest = dataset_copy.loc[dataset_copy["distance"].idxmin()]
        matches.append(
            f"Detected {tuple(color)} ≈ {closest['color_name']} "
            f"(RGB: {closest['R']},{closest['G']},{closest['B']})"
        )
    return matches

# -------- Main detection function --------
def detect_colors(image, clusters):
    if image is None:
        return ["No image provided."], "Upload an image first."

    img_np = np.array(image)
    img_np = img_np.reshape((-1, 3))  # flatten pixels

    try:
        kmeans = KMeans(n_clusters=clusters, random_state=42, n_init=10)
        kmeans.fit(img_np)
        colors = kmeans.cluster_centers_.astype(int)
    except Exception as e:
        return ["Error in KMeans"], str(e)

    # Detected RGB swatches
    swatches = [f"RGB: {tuple(color)}" for color in colors]

    # Dataset matches
    match_list = find_closest_colors(colors, labels_df)
    match_text = "\n".join(match_list)

    return swatches, match_text

# -------- Gradio UI --------
iface = gr.Interface(
    fn=detect_colors,
    inputs=[
        gr.Image(type="pil", label="Upload Image"),
        gr.Slider(1, 10, value=5, label="Number of Dominant Colors")
    ],
    outputs=[
        gr.Textbox(label="Detected RGB Colors"),
        gr.Textbox(label="Closest Dataset Matches")
    ],
    title="🎨 Color Detection from Images",
    description="Upload an image to detect dominant colors and match them with built-in color names."
)

iface.launch()
```

```
It looks like you are running Gradio on a hosted Jupyter notebook, which requires `share=True`. Automatically setting `share=True`

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://e53ed01e661033d62e.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the workin
```

🎨 Color Detection from Images

Upload an image to detect dominant colors and match them with built-in color names.

Upload Image



Detected RGB Colors

['RGB: (np.int64(170), np.int64(144), np.int64(201))', 'RGB: (np.int64(88), np.int64(39), np.int64(87))', 'RGB: (np.int64(24), np.int64(27), np.int64(22))', 'RGB: (np.int64(133), np.int64(80), np.int64(142))', 'RGB: (np.int64(225), np.int64(216), np.int64(242))']

Closest Dataset Matches

Detected (np.int64(170), np.int64(144), np.int64(201)) ≈ Silver (RGB: 192,192,192)
Detected (np.int64(88), np.int64(39), np.int64(87)) ≈ Purple (RGB: 128,0,128)
Detected (np.int64(24), np.int64(27), np.int64(22)) ≈ Black (RGB: 0,0,0)

Flag