import cv2

import numpy as np

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

image\_path = 'path\_to\_your\_uno\_card\_image.jpg'

image = cv2.imread(image\_path)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB) # Convert from BGR to RGB

image = load\_image(image\_path)

width=600:

# Resize image for faster processing

(h, w) = image.shape[:2]

aspect\_ratio = width / float(w)

height = int(h \* aspect\_ratio)

resized\_image=cv2.resize(image, (width, height))

# Get dominant color

K=4

# Reshape image to be a list of pixels

image = resized\_image.reshape((image.shape[0] \* resized\_image.shape[1], 3))

# Apply KMeans to find the dominant color

kmeans = KMeans(n\_clusters=k)

kmeans.fit(image)

# Get the most common color

counts = np.bincount(kmeans.labels\_)

dominant\_color = kmeans.cluster\_centers\_[np.argmax(counts)]

# Classify the detected color

rgb\_color= dominant\_color

# Simple color classification based on RGB values

red, green, blue = rgb\_color

if red > green and red > blue:

color\_name= "Red"

elif green > red and green > blue:

color\_name="Green"

elif blue > red and blue > green:

color\_name="Blue"

else:

color\_name=="Yellow" # Assuming UNO's 4 colors, the closest match for these

print(f"Detected color: {color\_name}")

# Plot the image

plt.imshow(resized\_image)

plt.axis('off')

plt.show()

import cv2

import numpy as np

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

# Load the image

image\_path = 'path\_to\_your\_uno\_card\_image.jpg'

image = cv2.imread(image\_path)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB) # Convert from BGR to RGB

# Resize the image

width = 600

(h, w) = image.shape[:2]

aspect\_ratio = width / float(w)

height = int(h \* aspect\_ratio)

resized\_image = cv2.resize(image, (width, height))

# Convert the RGB image to HSV color space

hsv\_image = cv2.cvtColor(resized\_image, cv2.COLOR\_RGB2HSV)

# Reshape the image to be a list of pixels

hsv\_image\_reshaped = hsv\_image.reshape((hsv\_image.shape[0] \* hsv\_image.shape[1], 3))

# Apply KMeans to find the dominant color in HSV space

k = 4 # Number of clusters

kmeans = KMeans(n\_clusters=k)

kmeans.fit(hsv\_image\_reshaped)

# Get the most common color

counts = np.bincount(kmeans.labels\_)

dominant\_color\_hsv = kmeans.cluster\_centers\_[np.argmax(counts)]

# Classify the color based on HSV values

hue, saturation, value = dominant\_color\_hsv

if hue < 15 or hue > 345:

color\_name = "Red"

elif 15 <= hue < 45:

color\_name = "Yellow"

elif 45 <= hue < 75:

color\_name = "Green"

elif 75 <= hue < 165:

color\_name = "Blue"

else:

color\_name = "Other"

print(f"Detected color: {color\_name}")

# Plot the original image

plt.imshow(resized\_image)

plt.axis('off')

plt.show()