Name: Seshasai.PB Reg_N0: 21MIA1005

Image and Video Analytics Assignment 1

Task 1: Basic Image Statistics and Color Space Conversion

Objective:

Compute basic statistics and convert an image into different color spaces.

Steps:

1. Read the Image: Load an image using OpenCV.

2. Compute Basic Statistics: Calculate the mean, standard deviation, and histogram of each color channel.

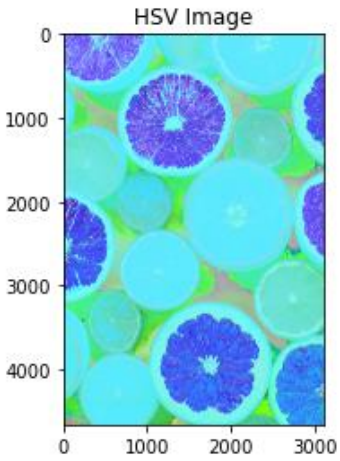
3. Convert Color Spaces: Convert the image to HSV and Lab color spaces and display the results.

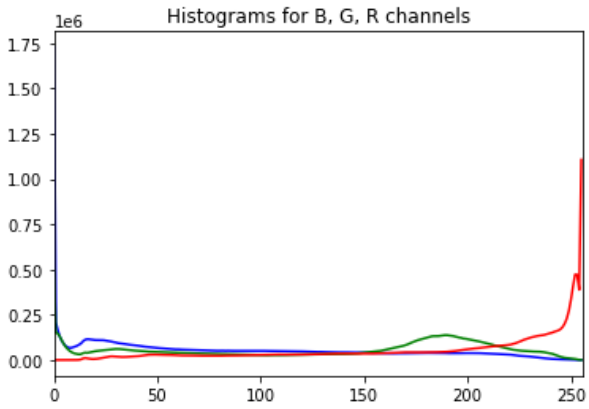
Code:

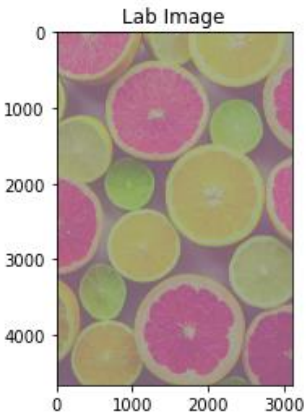
```
#Read the Image
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read the image
image = cv2.imread('img.jpg')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Display the image
plt.imshow(image_rgb)
plt.title('Original Image')
plt.show()

#Compute Basic Statistics
# Compute mean and standard deviation for each channel
means = cv2.mean(image)
std_devs = cv2.meanStdDev(image)[1]

print(f'Means: {means}')
print(f'Standard Deviations: {std_devs}')

# Compute and display histograms for each channel
colors = ('b', 'g', 'r')
for i, color in enumerate(colors):
    hist = cv2.calcHist([image], [i], None, [256], [0, 256])
    plt.plot(hist, color=color)
    plt.xlim([0, 256])
plt.title('Histograms for B, G, R channels')
plt.show()
#Convert Color Spaces
# Convert to HSV
```

```
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
hsv_image_rgb = cv2.cvtColor(hsv_image, cv2.COLOR_RGB2HSV)
plt.imshow(hsv_image_rgb)
plt.title('HSV Image')
plt.show()

# Convert to Lab
lab_image = cv2.cvtColor(image, cv2.COLOR_BGR2Lab)
lab_image_rgb = cv2.cvtColor(lab_image, cv2.COLOR_RGB2LAB)
plt.imshow(lab_image_rgb)
plt.title('Lab Image')
plt.show()
```

Plots:







Output:

Means: (81.08046014232826, 131.96532831209922, 193.0989555655932, 0.0)
Standard Deviations: [[70.26548136]
[76.57480909]
[66.14106548]]

Task 2: Simple Image Segmentation Using Thresholding

Objective:

Segment an image into foreground and background using global thresholding.

Steps:

1.Read the Image: Load a grayscale image.

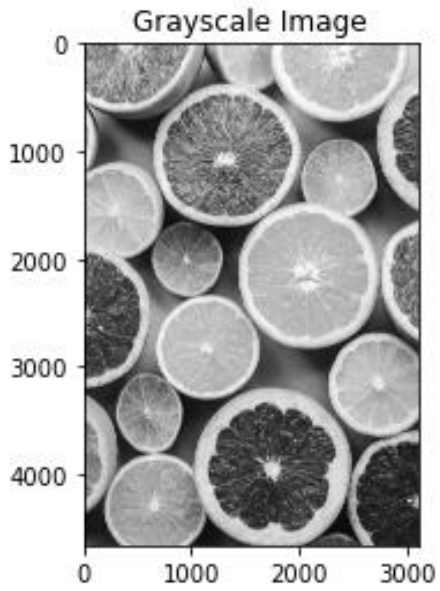
2.Apply Thresholding: Use a fixed threshold value to segment the image.

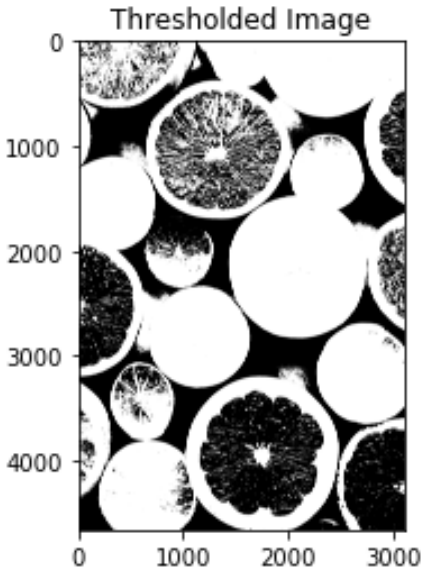
3.Display Results: Show the original and segmented images.

Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# Read the grayscale image
gray_image = cv2.imread('img.jpg', cv2.IMREAD_GRAYSCALE)
# Display the image
plt.imshow(gray_image, cmap='gray')
plt.title('Grayscale Image')
plt.show()
# Apply global thresholding
_, thresholded_image = cv2.threshold(gray_image, 127, 255,
cv2.THRESH_BINARY)
# Display the thresholded image
plt.imshow(thresholded_image, cmap='gray')
plt.title('Thresholded Image')
plt.show()
```

Plots:





Task 3: Color-Based Segmentation

Objective:

Segment specific objects in an image based on their color.

Steps:

1.Read the Image: Load an image with objects of different colors.

2.Convert to HSV: Convert the image to HSV color space.

3.Apply Color Thresholding: Use color thresholds to segment objects of a specific color.

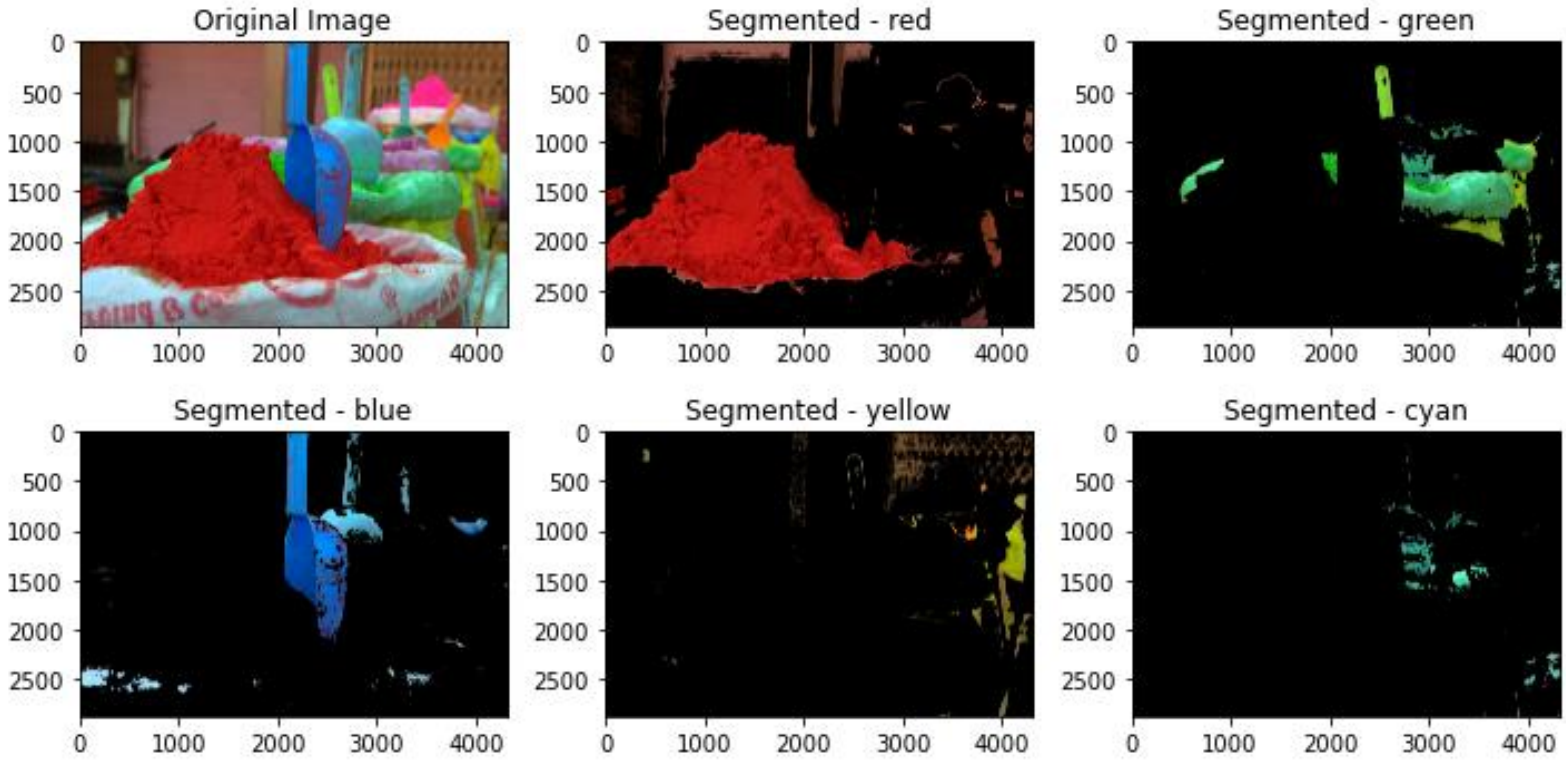
4.Display Results: Show the original and segmented images.

Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read the image
image = cv2.imread('img2.jpg')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Display the original image
plt.imshow(image_rgb)
plt.title('Original Image')
plt.show()

# Convert the image to HSV color space
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Define color ranges for segmentation
color_ranges = {
    'red': [(0, 120, 70), (10, 255, 255)],
    'green': [(36, 100, 100), (86, 255, 255)],
    'blue': [(94, 80, 2), (126, 255, 255)],
    'yellow': [(15, 100, 100), (35, 255, 255)],
    'cyan': [(78, 100, 100), (88, 255, 255)]
}

# Initialize a dictionary to store segmented images
segmented_images = {}

# Apply color thresholding for each color
for color, (lower_bound, upper_bound) in color_ranges.items():
    mask = cv2.inRange(hsv_image, np.array(lower_bound),
np.array(upper_bound))
    segmented_image = cv2.bitwise_and(image_rgb, image_rgb, mask=mask)
    segmented_images[color] = segmented_image

    # Display the segmented image for each color
```

```
    plt.imshow(segmented_image)
    plt.title(f'Segmented Image - {color}')
    plt.show()

# Display original and segmented images
plt.figure(figsize=(10, 5))
plt.subplot(2, 3, 1)
plt.imshow(image_rgb)
plt.title('Original Image')

for i, (color, segmented_image) in enumerate(segmented_images.items(),
start=2):
    plt.subplot(2, 3, i)
    plt.imshow(segmented_image)
    plt.title(f'Segmented - {color}')

plt.tight_layout()
plt.show()
```

Plots:



Task 4: Feature Extraction from Segmented Objects

Objective:

Extract and analyze features from segmented objects in an image.

Steps:

1.Read and Segment the Image: Load an image and segment objects based on color or intensity.

2.Extract Features: Calculate shape, color, and texture features for each segmented object (e.g., area, perimeter, mean color, texture metrics).

3.Analyze Features: Display and interpret the extracted features.

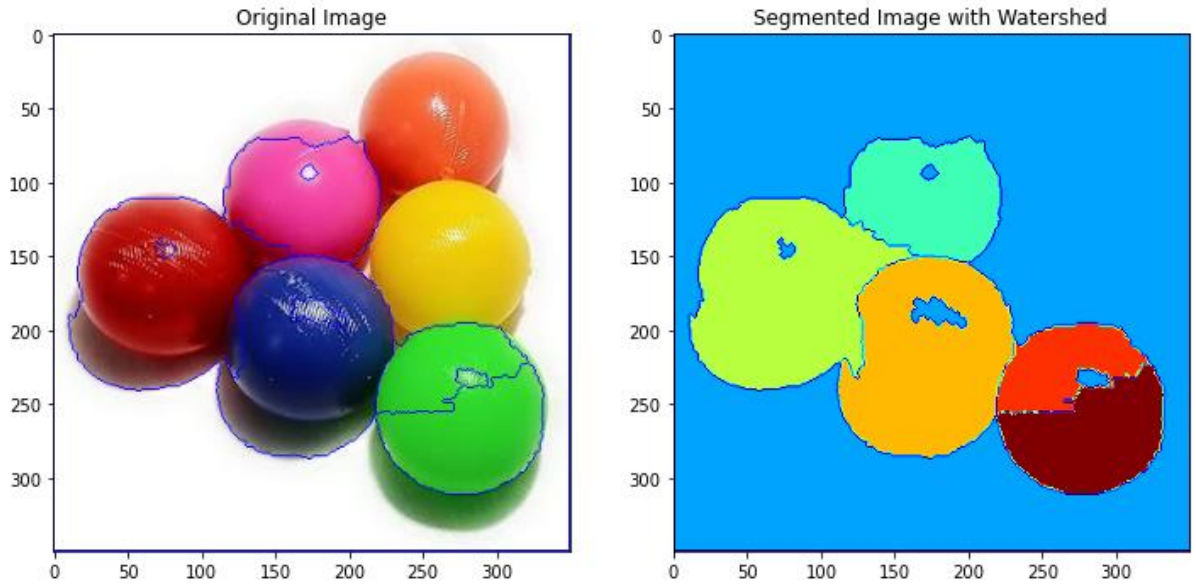
Code:

```
import cv2
import numpy as np
import pandas as pd
from skimage.measure import regionprops, label
import matplotlib.pyplot as plt

# Load the image
image_path = 'img4.jpg'
image = cv2.imread(image_path)
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Define color ranges for segmentation
color_ranges = {
    'orange': ([5, 50, 50], [15, 255, 255]),
    'yellow': ([20, 100, 100], [30, 255, 255]),
    'green': ([40, 50, 50], [70, 255, 255]),
    'blue': ([100, 50, 50], [140, 255, 255]),
    'red1': ([0, 50, 50], [10, 255, 255]),
    'red2': ([170, 50, 50], [180, 255, 255])
}

# Segment the image based on color ranges
segmented_images = {}
for color, (lower, upper) in color_ranges.items():
    lower_bound = np.array(lower, dtype="uint8")
    upper_bound = np.array(upper, dtype="uint8")
    mask = cv2.inRange(hsv_image, lower_bound, upper_bound)
    segmented_images[color] = mask

# Combine the two red masks
red_mask = segmented_images['red1'] | segmented_images['red2']
segmented_images['red'] = red_mask
del segmented_images['red1']
del segmented_images['red2']

# Extract features from the segmented objects
```

```
features = []
for color, mask in segmented_images.items():
    # Perform noise removal and watershed algorithm
    kernel = np.ones((3, 3), np.uint8)
    opening = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel,
iterations=2)
    sure_bg = cv2.dilate(opening, kernel, iterations=3)
    dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
    ret, sure_fg = cv2.threshold(dist_transform, 0.7 *
dist_transform.max(), 255, 0)
    sure_fg = np.uint8(sure_fg)
    unknown = cv2.subtract(sure_bg, sure_fg)
    ret, markers = cv2.connectedComponents(sure_fg)
    markers = markers + 1
    markers[unknown == 255] = 0
    markers = cv2.watershed(image, markers)
    image[markers == -1] = [255, 0, 0]

    # Extract features
    labeled_image = label(markers > 1)
    regions = regionprops(labeled_image)
    for region in regions:
        features.append({
            'Color': color,
            'Area': region.area,
            'Perimeter': region.perimeter,
            'Eccentricity': region.eccentricity,
            'Solidity': region.solidity
        })

features_df = pd.DataFrame(features)

# Count the number of segmented objects
object_count = len(features_df)

# Display the results
print(features_df)

# Save DataFrame to a CSV file
features_df.to_csv('colored_segmented_features.csv', index=False)

# Display the original and segmented images
plt.figure(figsize=(12, 6))
plt.subplot(121)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.subplot(122)
plt.imshow(markers, cmap='jet')
```

```
plt.title('Segmented Image with Watershed')
plt.show()
```

Plots:



Outputs:

Numbers of Objects found by Contours (**a curve joining all the continuous points (along the boundary), having same color or intensity**.) is 5.

```
In [1]: runfile('D:/Sem7/Image and video analytics/Lab/lab1/
Lab4.py', wdir='D:/Sem7/Image and video analytics/Lab/lab1')
    Color     Area   Perimeter  Eccentricity  Solidity
0  orange   6173.0  509.587878      0.649558  0.903807
1  yellow   8506.0  363.847763      0.329070  0.954015
2   green  10676.0  409.439646      0.159020  0.972490
3    blue   8812.0  475.730014      0.419275  0.918299
4     red  12847.0  500.759451      0.554432  0.907851
```

Task 5: Multi-Object Segmentation and Counting

Objective:

Segment multiple objects in an image and count them.

Steps:

1.Read the Image: Load an image containing multiple objects.

2.Preprocess the Image: Apply preprocessing techniques such as filtering and color space conversion.

3.Segment Objects: Use techniques like thresholding, contour detection, or clustering to segment the objects.

4.Count Objects: Identify and count the number of segmented objects.

5.Display Results: Show the original image with the segmented objects highlighted and the count of objects.

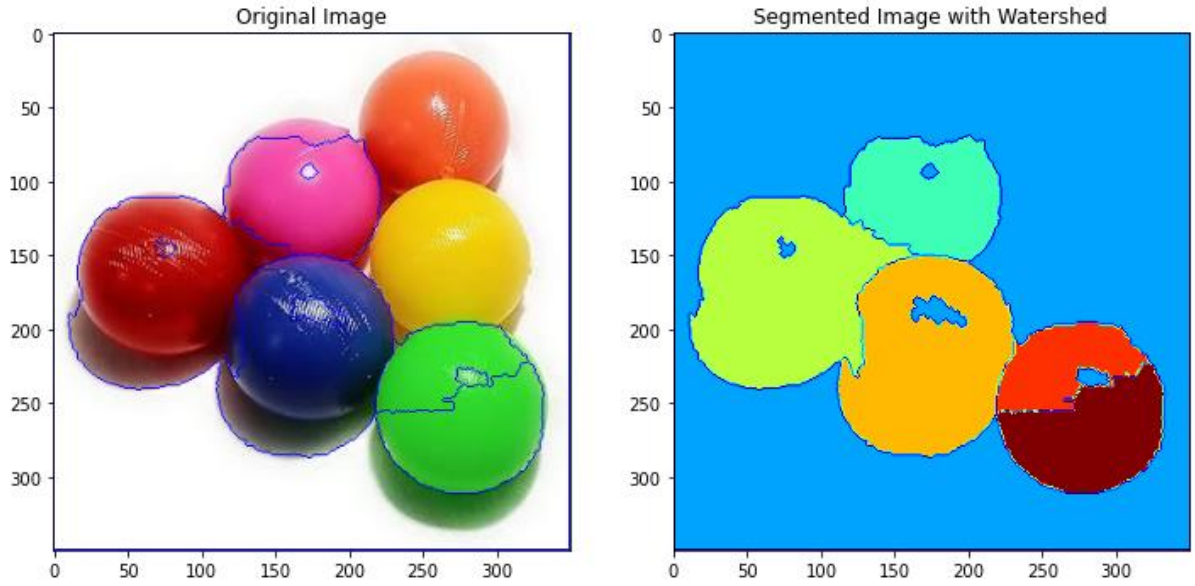
Code:

```
import cv2
import numpy as np
import pandas as pd
from skimage.measure import regionprops, label
import matplotlib.pyplot as plt

# Load the image
image_path = 'path_to_your_image.jpg'
image = cv2.imread(image_path)
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Define color ranges for segmentation
color_ranges = {
    'orange': ([5, 50, 50], [15, 255, 255]),
    'yellow': ([20, 100, 100], [30, 255, 255]),
    'green': ([40, 50, 50], [70, 255, 255]),
    'blue': ([100, 50, 50], [140, 255, 255]),
    'red1': ([0, 50, 50], [10, 255, 255]),
    'red2': ([170, 50, 50], [180, 255, 255])
}

# Segment the image based on color ranges
segmented_images = {}
for color, (lower, upper) in color_ranges.items():
    lower_bound = np.array(lower, dtype="uint8")
    upper_bound = np.array(upper, dtype="uint8")
    mask = cv2.inRange(hsv_image, lower_bound, upper_bound)
    segmented_images[color] = mask

# Combine the two red masks
red_mask = segmented_images['red1'] | segmented_images['red2']
segmented_images['red'] = red_mask
```

```
del segmented_images['red1']
del segmented_images['red2']

# Extract features from the segmented objects
features = []
for color, mask in segmented_images.items():
    # Perform noise removal and watershed algorithm
    kernel = np.ones((3, 3), np.uint8)
    opening = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel,
iterations=2)
    sure_bg = cv2.dilate(opening, kernel, iterations=3)
    dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
    ret, sure_fg = cv2.threshold(dist_transform, 0.7 *
dist_transform.max(), 255, 0)
    sure_fg = np.uint8(sure_fg)
    unknown = cv2.subtract(sure_bg, sure_fg)
    ret, markers = cv2.connectedComponents(sure_fg)
    markers = markers + 1
    markers[unknown == 255] = 0
    markers = cv2.watershed(image, markers)
    image[markers == -1] = [255, 0, 0]

    # Extract features
    labeled_image = label(markers > 1)
    regions = regionprops(labeled_image)
    for region in regions:
        features.append({
            'Color': color,
            'Area': region.area,
            'Perimeter': region.perimeter,
            'Eccentricity': region.eccentricity,
            'Solidity': region.solidity
        })

features_df = pd.DataFrame(features)

# Count the number of segmented objects
object_count = len(features_df)

# Display the results
print(f"Number of segmented objects: {object_count}")
print(features_df)

# Save DataFrame to a CSV file
features_df.to_csv('/mnt/data/colored_segmented_features.csv',
index=False)

# Display the original and segmented images
```

```
plt.figure(figsize=(12, 6))
plt.subplot(121)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.subplot(122)
plt.imshow(markers, cmap='jet')
plt.title('Segmented Image with Watershed')
plt.show()
```

Plots:



Output:

```
In [21]: runfile('D:/Sem7/Image and video analytics/Lab/lab1/Lab5.py',
wdir='D:/Sem7/Image and video analytics/Lab/lab1')
Number of segmented objects: 5
    Color     Area   Perimeter  Eccentricity  Solidity
0  orange   6173.0  509.587878      0.649558  0.903807
1  yellow   8506.0  363.847763      0.329070  0.954015
2   green  10676.0  409.439646      0.159020  0.972490
3    blue   8812.0  475.730014      0.419275  0.918299
4     red  12847.0  500.759451      0.554432  0.907851
```