Jaypee University Of Information Technology
Solan, Himachal Pradesh

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Machine Learning Lab(18B1WCL674)

Project Name: Most Dominant Color

Submitted to

Dr. Nancy Singla
Assistant Professor (SG)

Submitted by

Vishesh Jaiswal
211369
Anuroop Srivastava
211398

INTRODUCTION

1.1 Project Overview

The objective of this project is to develop a machine learning model that can analyze an image and determine the most dominating color. This capability can be applied in various fields such as digital art, fashion, interior design, and image editing software.
"Finding Most Dominant Color in an Image Using K-Means Algorithm" explores image processing and machine learning to develop a method for extracting essential color information from digital photos. Using the K-Means clustering algorithm, the study groups pixels with similar color values to identify the most prevalent colors in an image. This project has significant implications for fields such as graphic design, image processing, and text analysis. The primary goal is to automate color extraction, enhancing tasks like object recognition, image classification, and visual content creation. The study demonstrates how machine learning can simplify image processing workflows, paving the way for advancements in visual-based applications.

Understanding an image's color composition is crucial for various applications, including image segmentation, content-based image retrieval, and color-based object detection. In the digital age, analyzing visual content has become increasingly important. Color detection, which identifies and quantifies the colors in an image, is particularly useful in digital art, design, marketing, and content curation.

1.2 Problem Statement

Given an image, we need to identify the color that occupies the largest area. This requires processing the image data to segment and quantify colors, and then determine which color is the most prevalent.

OBJECTIVE

The primary objective of this project is to develop a machine learning model capable of accurately identifying the most dominating color in any given image. This involves several key goals:

1.Image Analysis:

- To preprocess images to standardize them for analysis.
- To convert image color data into a suitable format for processing.

2. Color Quantization:

- To reduce the number of colors in an image while retaining the essential color information.
- To convert color data to a perceptually uniform color space (e.g., RGB) for better clustering performance.

3. Clustering and Identification:

- To apply clustering algorithms, specifically K-Means, to group similar colors within the image.
- To identify the cluster (color) with the highest number of pixels, thereby determining the most dominating color.

4. Model Evaluation:

- To evaluate the performance of the clustering algorithm in accurately identifying the most dominating color.

5. Practical Application:

- To develop a functional tool or script that can be used to find the most dominating color in any image.
- To explore potential applications of this tool in various fields such as digital art, fashion, interior design, and marketing.

METHODOLOGY

1. **Image Acquisition:** Obtain the target image for color dominance analysis. The image should be in a format compatible with the OpenCV library (commonly JPEG or PNG).
2. **Image Resizing:** Use the imutils library to resize the image to a specified height while maintaining the aspect ratio. This step ensures uniformity in the analysis and reduces computational complexity.
3. **Flatten Image:** Flatten the resized image to create a feature vector. Reshape the image to a 2D array where each row represents a pixel and each column represents the RGB values
4. **K-Means Clustering:** Apply the K-Means clustering algorithm to the flattened image. Choose the number of clusters based on the desired level of color granularity. The scikit-learn library provides the K-Means class for this purpose.
5. **Dominant Colors Extraction:** Retrieve the cluster centers, which represent the dominant colors in the image. These centers are the average color values of pixels assigned to each cluster during the clustering process.
6. **Color Proportions Calculation:** Calculate the proportion of each dominant color in the image by computing the percentage of pixels

assigned to each cluster. This information is crucial for understanding the color composition.

7. **Visualization:** Utilize Matplotlib to create visualizations of the dominant colors. This may include subplots displaying each dominant color and a bar chart illustrating the proportions of each color.
8. **Image Overlay:** Resize the original image to a desired display size, maintaining the aspect ratio. Create a copy of the resized image and overlay it with a white rectangle to display information about the most dominant colors.
9. **Final Visualization:** Combine the original image with the overlaid information, creating a final visualization that highlights the dominant colors. Add text annotations to convey relevant details about the color analysis.
10. **Display and Save Results:** Display the final visualization using OpenCV. Optionally, save the resulting image to a file (e.g., 'output.jpg') for future reference.

TOOLS AND TECHNOLOGY

1. OpenCV (cv2): Used for image processing tasks such as reading, resizing, drawing rectangles, adding text, and displaying images.

2. NumPy (np): Utilized for numerical operations, particularly for reshaping arrays and mathematical computations on images.

3. Matplotlib (plt): Used for plotting graphs and displaying images in the form of blocks and bar charts.

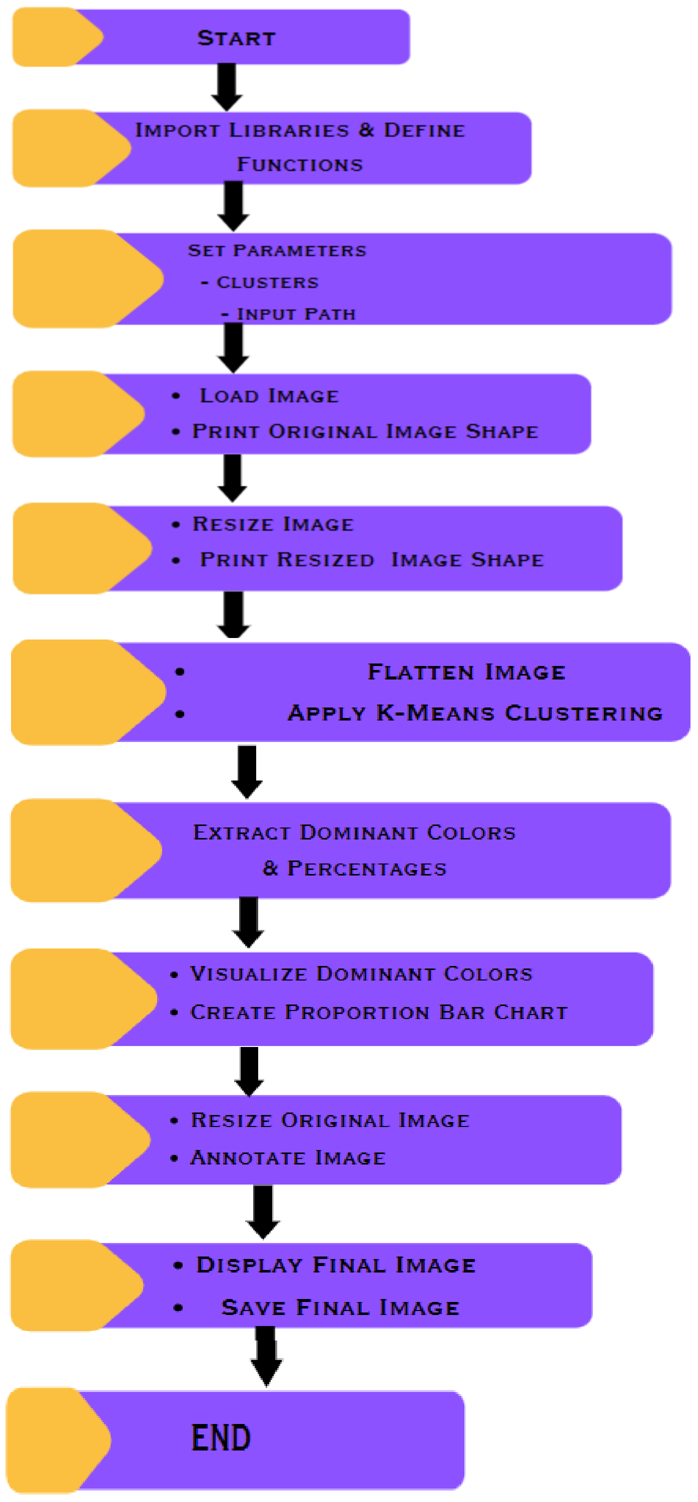
4. Scikit-learn(sklearn): Specifically, the KMeans class from scikit-learn is used for performing K-means clustering on the colors extracted from the image.

5. Imutils: A set of convenience functions to make basic image processing functions such as resizing, rotating, and displaying images easier with OpenCV.

6. Web Colors: Specifically, used for converting RGB values to color names based on the CSS3 specification.

These tools and libraries collectively facilitate the image processing, color analysis, visualization, and manipulation tasks performed in the code.

WORKFLOW DIAGRAM

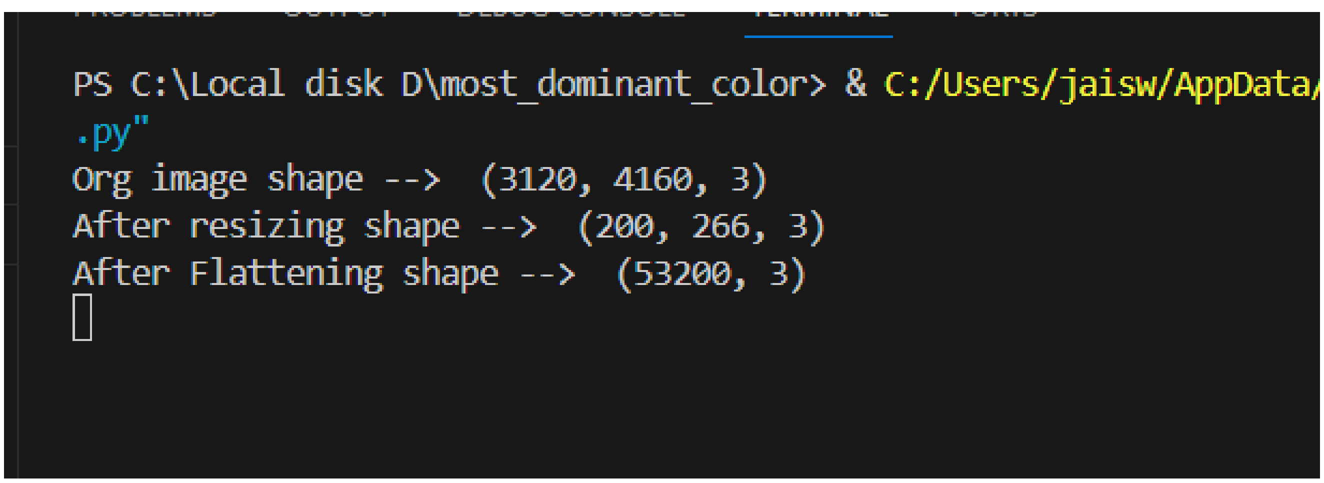

Start
Import Libraries & Define Functions
Set Parameters
- Clusters
- Input Path
• Load Image
• Print Original Image Shape
• Resize Image
• Print Resized Image Shape
• Flatten Image
• Apply K-Means Clustering
Extract Dominant Colors & Percentages
• Visualize Dominant Colors
• Create Proportion Bar Chart
• Resize Original Image
• Annotate Image
• Display Final Image
• Save Final Image
END

IMPLEMENTATION

```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import imutils
from webcolors import rgb_to_name, CSS3_HEX_TO_NAMES

def get_color_name(rgb_tuple):
    min_colors = {}
    for key, name in CSS3_HEX_TO_NAMES.items():
        r_c, g_c, b_c = hex_to_rgb(key)
        rd = (r_c - rgb_tuple[0]) ** 2
        gd = (g_c - rgb_tuple[1]) ** 2
        bd = (b_c - rgb_tuple[2]) ** 2
        min_colors[(rd + gd + bd)] = name
    return min_colors[min(min_colors.keys())]

def hex_to_rgb(hex):
    hex = hex.lstrip('#')
    hlen = len(hex)
    return tuple(int(hex[i:i+hlen//3], 16) for i in range(0, hlen,
hlen//3))

clusters = 5 # try changing it

input_path = 'C:\\Local disk
D\\most_dominant_color\\input_images\\IMG_3.jpg'
img = cv2.imread(input_path)
org_img = img.copy()
print('Org image shape --> ', img.shape)

img = imutils.resize(img, height=200)
print('After resizing shape --> ', img.shape)

flat_img = np.reshape(img, (-1, 3))
print('After Flattening shape --> ', flat_img.shape)
```

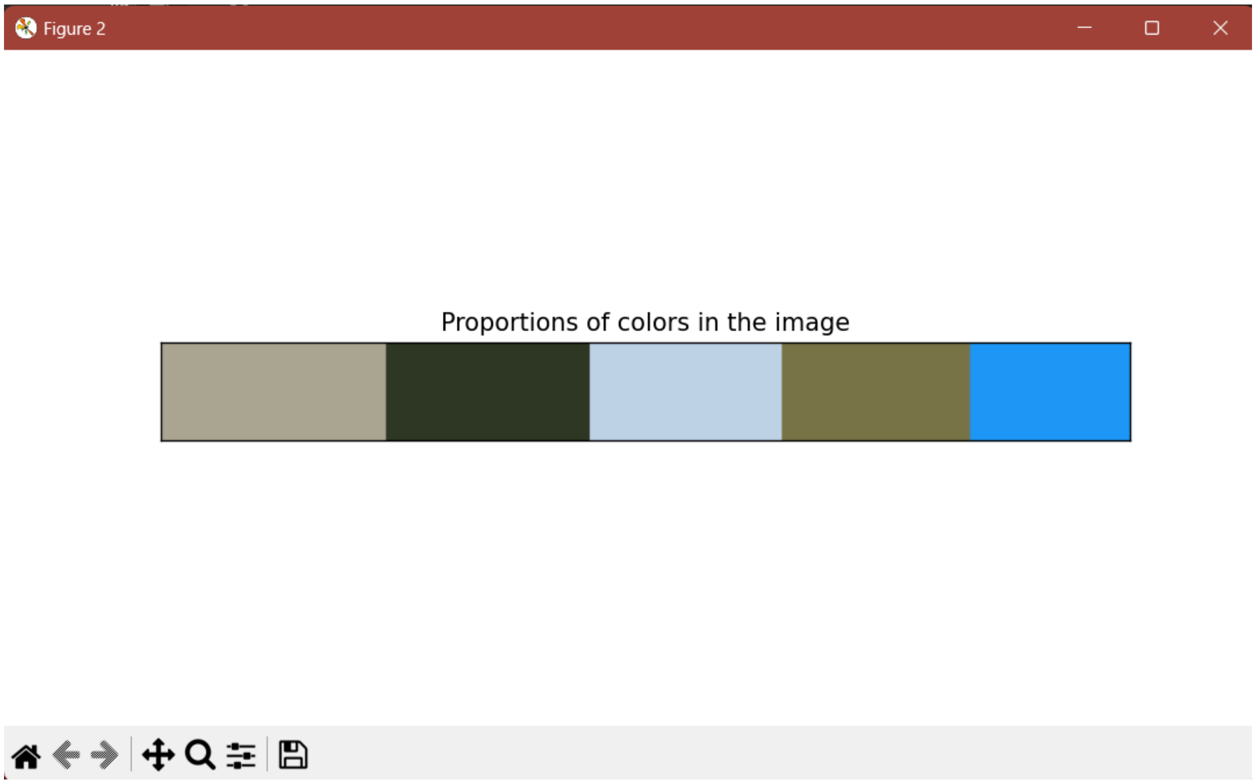
```python
kmeans = KMeans(n_clusters=clusters, random_state=0)
kmeans.fit(flat_img)

dominant_colors = np.array(kmeans.cluster_centers_, dtype='uint')

percentages = (np.unique(kmeans.labels_, return_counts=True)[1]) /
flat_img.shape[0]
p_and_c = zip(percentages, dominant_colors)
p_and_c = sorted(p_and_c, reverse=True)

# Display color blocks with percentages and names
block = np.ones((50, 50, 3), dtype='uint')
plt.figure(figsize=(12, 8))
for i in range(clusters):
    plt.subplot(1, clusters, i + 1)
    block[:] = p_and_c[i][1][::-1]  # convert BGR to RGB
    plt.imshow(block)
    plt.xticks([])
    plt.yticks([])
    color_name = get_color_name(p_and_c[i][1][::-1])
    plt.xlabel(f"{color_name}\n{round(p_and_c[i][0] * 100, 2)}%")

# Display bar chart
bar = np.ones((50, 500, 3), dtype='uint')
plt.figure(figsize=(12, 8))
plt.title('Proportions of colors in the image')
start = 0
i = 1
for p, c in p_and_c:
    end = start + int(p * bar.shape[1])
    if i == clusters:
        bar[:, start:] = c[::-1]
    else:
        bar[:, start:end] = c[::-1]
    start = end
    i += 1

plt.imshow(bar)
plt.xticks([])
plt.yticks([])

rows = 1000
cols = int((org_img.shape[0] / org_img.shape[1]) * rows)
img = cv2.resize(org_img, dsize=(rows, cols),
interpolation=cv2.INTER_LINEAR)

copy = img.copy()
cv2.rectangle(copy, (rows // 2 - 250, cols // 2 - 90), (rows // 2 + 250,
cols // 2 + 110), (255, 255, 255), -1)

final = cv2.addWeighted(img, 0.1, copy, 0.9, 0)
cv2.putText(final, 'Most Dominant Colors in the Image', (rows // 2 - 230,
cols // 2 - 40), cv2.FONT_HERSHEY_DUPLEX, 0.8, (0, 0, 0), 1, cv2.LINE_AA)

start = rows // 2 - 220
for i in range(5):
    end = start + 70
    final[cols // 2:cols // 2 + 70, start:end] = p_and_c[i][1]
    cv2.putText(final, str(i + 1), (start + 25, cols // 2 + 45),
cv2.FONT_HERSHEY_DUPLEX, 1, (255, 255, 255), 1, cv2.LINE_AA)
    start = end + 20

plt.show()

cv2.imshow('img', final)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Add "output" prefix to the original image name
output_dir = 'C:\\Local disk D\\most_dominant_color\\output_images\\'
base_name = os.path.basename(input_path)
output_name = f"output_{base_name}"
output_path = os.path.join(output_dir, output_name)

cv2.imwrite(output_path, final)
```

RESULT

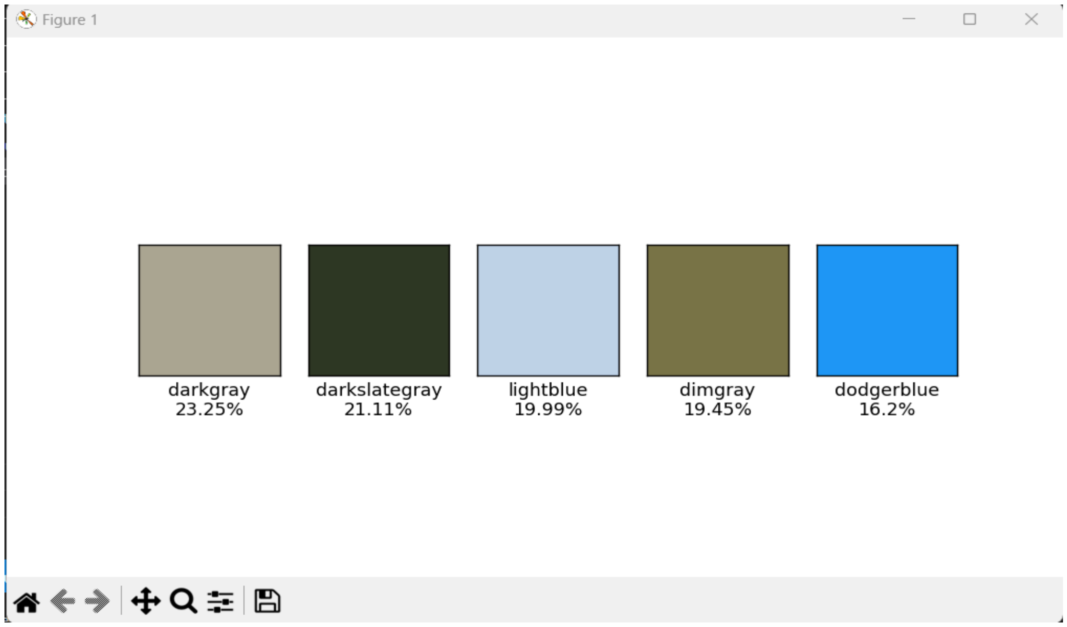


```
PS C:\Local disk D\most_dominant_color> & C:/Users/jaisw/AppData/
.py"
Org image shape -->  (3120, 4160, 3)
After resizing shape -->  (200, 266, 3)
After Flattening shape -->  (53200, 3)
```

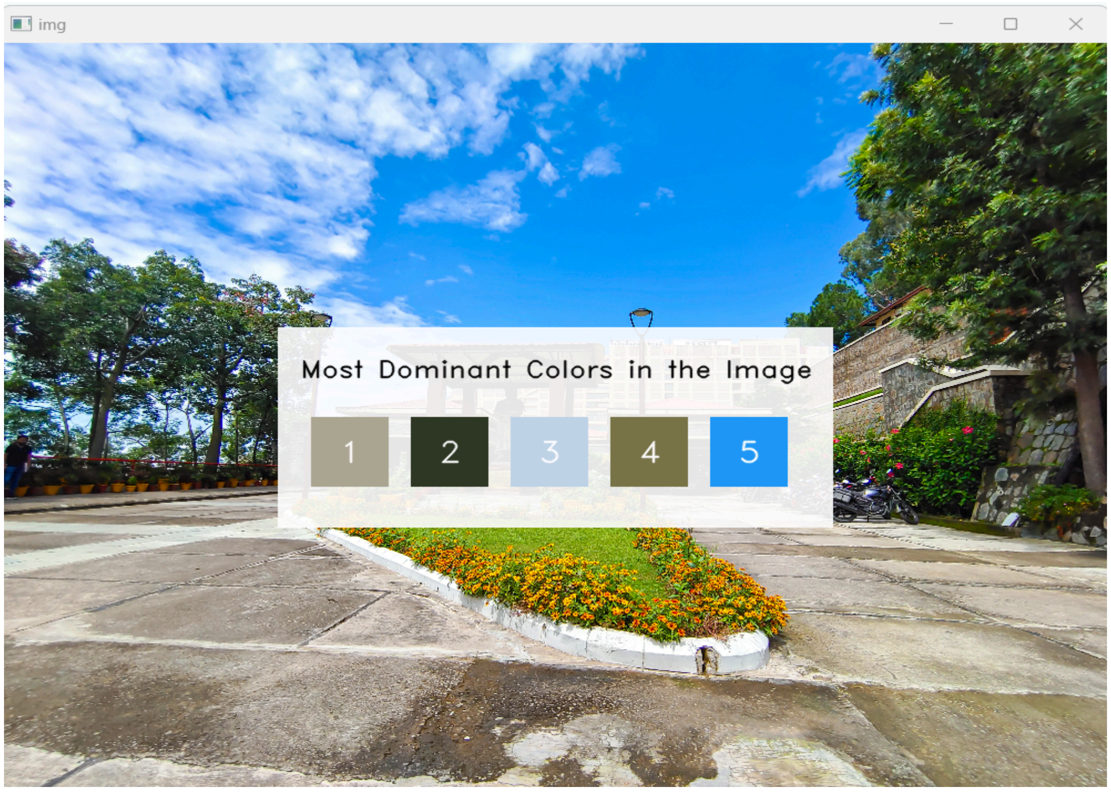
Dimensions of processing images



Proportion of color in the image



<u>Percentage of color in image</u>



<u>Output Image</u>

Conclusion

The implemented code successfully identifies and visualizes the most dominant colors present in an input image. By leveraging various tools and technologies, including OpenCV for image processing, Scikit-learn for clustering, NumPy for data manipulation, Matplotlib for visualization, and web colors for color naming, the code achieves the project objective effectively.

Key Achievements

1. Color Identification: The code accurately clusters the colors in the input image using the KMeans clustering algorithm, determining the most dominant colors.

2. Visualization: The code provides visualizations of the dominant colors through color blocks accompanied by their corresponding percentages and names, as well as a bar chart illustrating the proportions of each color.

3. Image Integration: The code seamlessly integrates the color visualization with the original image, overlaying the dominant colors on top of the image and annotating them with their respective ranks.

Limitations and Future Improvements

1. Cluster Sensitivity: The performance of the KMeans algorithm may vary depending on the number of clusters chosen. Experimentation with different cluster values could improve accuracy.

2. Color Naming Accuracy: While the code attempts to name colors using web color names, the accuracy of color naming may vary, especially for shades and variations.

3. User Interface: The code currently lacks a user-friendly interface for interacting with the tool. Integrating it into a graphical user interface (GUI) could enhance usability.

Application Potential

1. Digital Art and Design: Designers and artists can use this tool to identify dominant colors in images for inspiration or color palette creation.

2. Fashion and Interior Design: Professionals in fashion and interior design can utilize the tool for color matching and trend analysis.

3. Marketing and Branding: Marketers and brand managers can employ the tool to analyze brand images and ensure consistency in color representation across marketing materials.

Conclusion Statement

In conclusion, the developed code provides a robust solution for identifying and visualizing the most dominant colors in an image. With further refinement and integration into user-friendly interfaces, the tool holds significant potential for various applications in creative, design, and marketing domains.

REFERENCES

1. Dominant colors in an image using k-means clustering

   Link:
   https://medium.com/buzzrobot/dominant-colors-in-an-image-using-k-means-clustering-3c7af4622036

2. Everything you need to know about K-Means Clustering

   Link:
   https://medium.com/analytics-vidhya/everything-you-need-to-know-about-k-means-clustering-88ad4058cce0

3. Finding the Most Dominant Color in an Image using K-Means Algorithm Based on Machine Learning

   Link:
   https://www.ijraset.com/research-paper/finding-the-most-dominant-color-in-an-image