THE SPARKS FOUNDATION - COMPUTER VISION AND IOT INTERNSHIP

GRIP_MARCH-2022

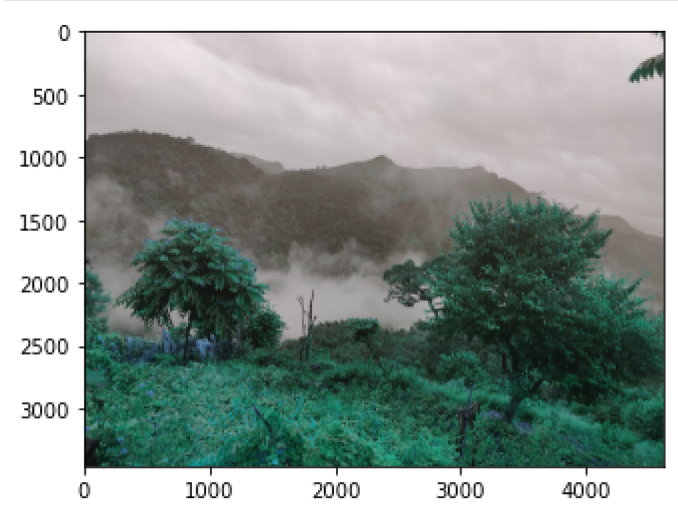
NAME: NOTAM KEDARI

Importing The Required Libraries

In [3]:
```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
from sklearn.cluster import KMeans
from skimage.color import rgb2lab, deltaE_cie76
```

Reading image

In [5]:
```
image = cv2.imread('C:/Users/NOTAM KEDARI/Desktop/sample_image.jpg')
plt.imshow(image)
plt.show()
```



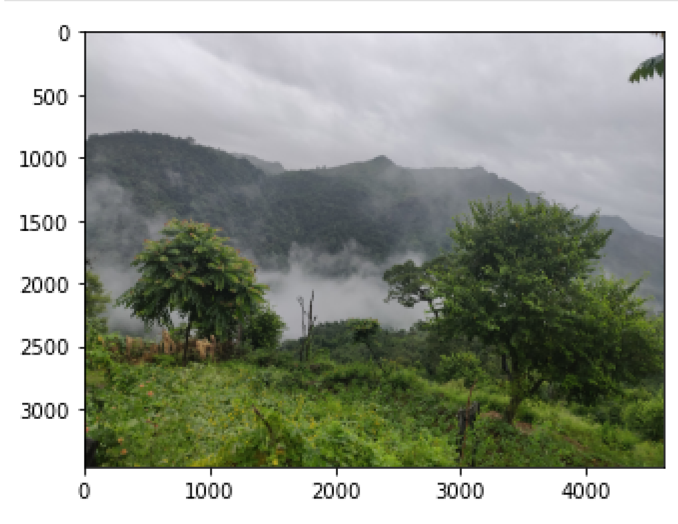
Checking Type and Shape of image Data

In [6]:
```
print("The type of this input is {}".format(type(image)))
print("Shape: {}".format(image.shape))
```

```
The type of this input is <class 'numpy.ndarray'>
Shape: (3456, 4608, 3)
```

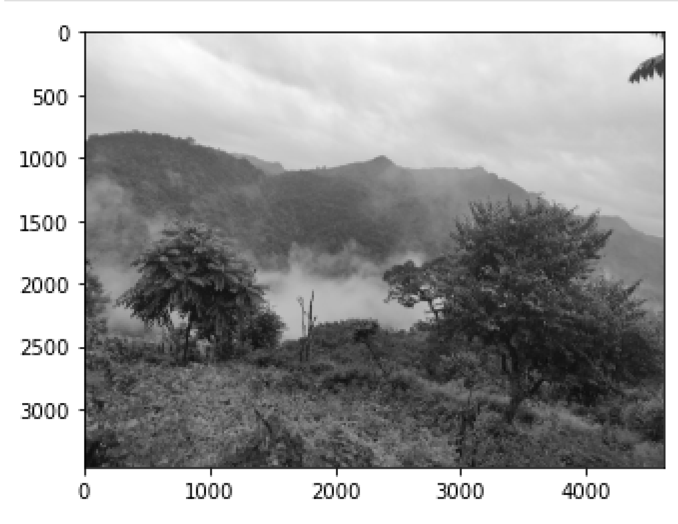
Converting BGR-To-RGB

In [7]:
```
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.imshow(image)
plt.show()
```



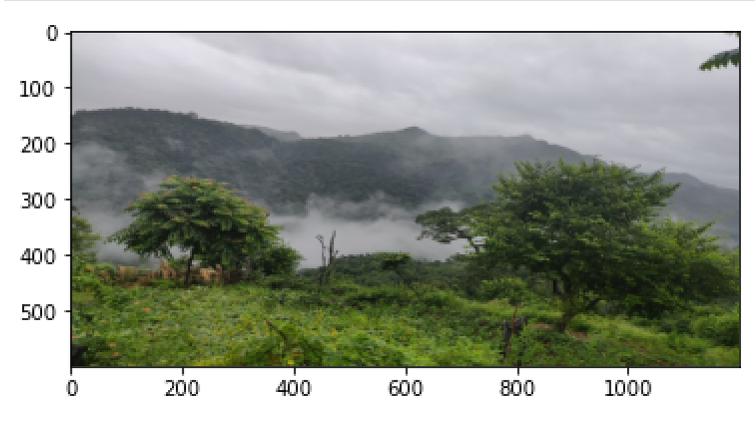
Converting Image to GrayScale

In [8]:
```
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.imshow(gray_image, cmap='gray')
plt.show()
```



Resizing Images

In [9]:
```
resized_image = cv2.resize(image, (1200, 600))
plt.imshow(resized_image)
plt.show()
```



COLOUR IDENTIFICATION

In [10]:
```
def RGB2HEX(color):
    return "#{:02x}{:02x}{:02x}".format(int(color[0]), int(color[1]), int(color[2]))
```

Defining method for getting images & Conversion from BGR-TO-RGB

In [11]:
```
def get_image(image_path):
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return image
```

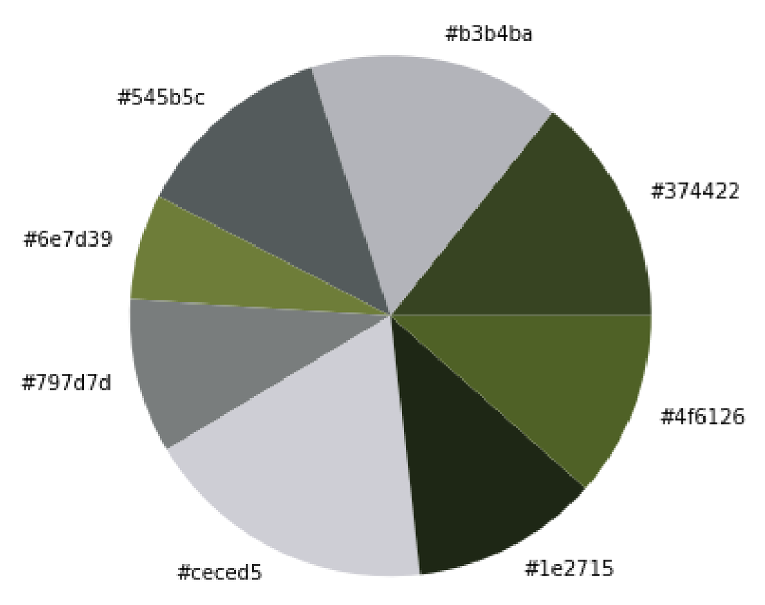
Defining method along with K_Means Algorithm

In [12]:
```
def get_colors(image, number_of_colors, show_chart):

    modified_image = cv2.resize(image, (600, 400), interpolation = cv2.INTER_AREA)
    modified_image = modified_image.reshape(modified_image.shape[0]*modified_image.shape[1], 3)

    clf = KMeans(n_clusters = number_of_colors)
    labels = clf.fit_predict(modified_image)

    counts = Counter(labels)
    counts = dict(sorted(counts.items()))

    center_colors = clf.cluster_centers_
    ordered_colors = [center_colors[i] for i in counts.keys()]

    hex_colors = [RGB2HEX(ordered_colors[i]) for i in counts.keys()]
    rgb_colors = [ordered_colors[i] for i in counts.keys()]

    if (show_chart):
        plt.figure(figsize = (8, 6))
        plt.pie(counts.values(), labels = hex_colors, colors = hex_colors)

    return rgb_colors
```

Calling Above Two Functions inside each other

In [14]:
```
get_colors(get_image('C:/Users/NOTAM KEDARI/Desktop/sample_image.jpg'), 8, True)
```

Out[14]:
```
[array([55.46719566, 68.4372648 , 34.54832405]),
 array([179.7869794 , 180.35941549, 186.81486923]),
 array([84.25194044, 91.09117559, 92.33288917]),
 array([110.49828789, 125.91369689,  57.1324033 ]),
 array([121.20818343, 125.75649083, 125.70655802]),
 array([206.76034888, 206.45446035, 213.92752928]),
 array([30.69521195, 39.211594  , 21.41541588]),
 array([79.4193677 , 97.26212747, 38.69287941])]
```



Search images using Color

In [20]:
```
IMAGE_DIRECTORY = 'C:/Users/NOTAM KEDARI/Desktop/images'

COLORS = {'GREEN': [0, 128, 0],'BLUE': [0, 0, 128],'YELLOW': [255, 255, 0]}

images = []

for file in os.listdir(IMAGE_DIRECTORY):
    if not file.startswith('.'):
        images.append(get_image(os.path.join(IMAGE_DIRECTORY, file)))
```

Visualization of Data

In [21]:
```
plt.figure(figsize=(20, 5))
for i in range(len(images)):
    plt.subplot(1, len(images), i+1)
    plt.imshow(images[i])
    plt.axis('off')
```

Function for finding matches by using top 10 colors in images

In [22]:
```
def match_image_by_color(image, color, threshold = 60, number_of_colors = 10):

    image_colors = get_colors(image, number_of_colors, False)
    selected_color = rgb2lab(np.uint8(np.asarray([[color]])))

    select_image = False

    for i in range(number_of_colors):

        curr_color = rgb2lab(np.uint8(np.asarray([[image_colors[i]]])))
        diff = deltaE_cie76(selected_color, curr_color)

        if (diff < threshold):
            select_image = True

    return select_image
```

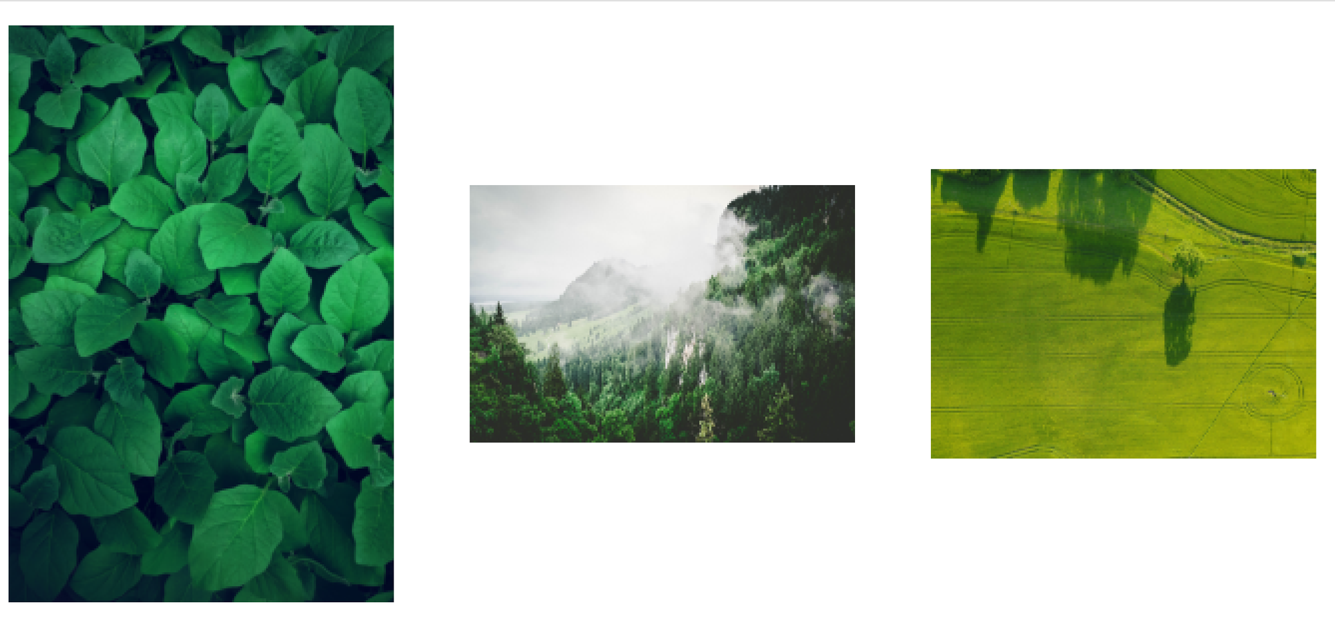
Function for selection of images

In [23]:
```
def show_selected_images(images, color, threshold, colors_to_match):
    index = 1

    for i in range(len(images)):
        selected = match_image_by_color(images[i],
                                        color,
                                        threshold,
                                        colors_to_match)
        if (selected):
            plt.subplot(1, 5, index)
            plt.imshow(images[i])
            plt.axis('off')
            index += 1
```

Calling above methods and visualizing results

FINDING GREEN COLOR

In [24]:
```
plt.figure(figsize = (20, 8))
show_selected_images(images, COLORS['GREEN'], 60, 5)
```

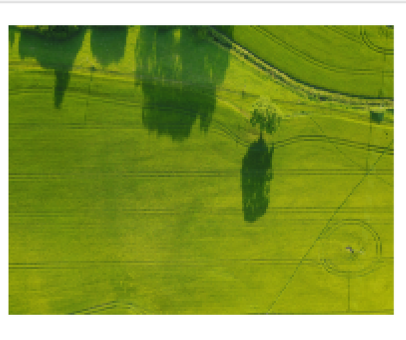


In [ ]:
```
FINDING BLUE COLOR
```

In [25]:
```
plt.figure(figsize = (20, 10))
show_selected_images(images, COLORS['BLUE'], 60, 5)
```

FINDING YELLOW COLOUR

In [27]:
```
plt.figure(figsize = (20, 10))
show_selected_images(images, COLORS['YELLOW'], 60, 5)
```



DONE BY: NOTAM KEDARI