====K-Means Clustering for Image====#

In [1]:

```
'''Load the Required Packages'''
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from skimage import io
from ipywidgets import interact, widgets
from IPython.display import display
#!pip install opencv-python
import cv2 as cv
import skfuzzy as fuzz
from skimage.color import rgb2gray
```

In [2]:

```
# Load the image
image=cv.imread('E:/Village of Study/STAT - 811 IMAGE CLASSIFICATION/HABIB JU PROFILE IMAGE.jpg')
# Convert to RGB (if needed)
image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
# Check the image shape of Color Image
print(image.shape)
```

```
(200, 200, 3)
```

In [3]:

```
'''Converts the MxNx3 image into a Kx3 matrix where K=MxN
and each row is now a vector in the 3-D space of RGB'''

c_pixels = image.reshape(-1, 3)
c_pixels

'''Convert the unit8 values to float as it is a requirement
   of the k-means method of OpenCV'''

c_pixels = np.float32(c_pixels)
c_pixels
```

Out[3]:

```
array([[255., 255., 255.],
       [255., 255., 255.],
       [255., 255., 255.],
       ...,
       [255., 255., 255.],
       [255., 255., 255.],
       [255., 255., 255.]], dtype=float32)
```

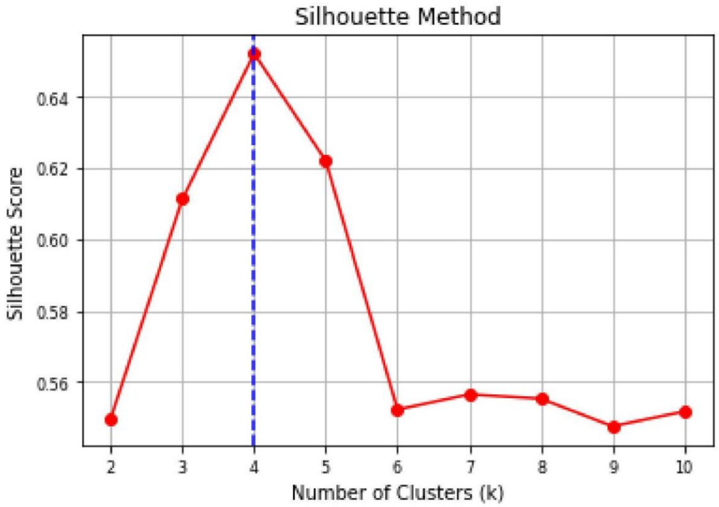
In [6]:

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

In [7]:

```
silhouette_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(c_pixels)
    silhouette_avg = silhouette_score(c_pixels, labels)
    silhouette_scores.append(silhouette_avg)
```

In [8]:
```
plt.figure(figsize=(6, 4))
plt.plot(range(2, 11), silhouette_scores, marker='o',color='red')
plt.axvline(x=4, color='b',ls='--',linewidth=1.5)
plt.title('Silhouette Method')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Silhouette Score')
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.grid()
plt.show()
```



In [9]:
```
'''Define criteria, number of clusters(K) and apply k-means()'''
'''Specify the algorithm's termination criteria'''
criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 10, 1.0)
criteria
```

Out[9]:
```
(3, 10, 1.0)
```

In [10]:
```
'''Compactness is the Sum of Squared distance from each point to their Corresponding Centers'''
np.random.seed(104729)
WSS = []
K=[]
for k in range(2, 11):
    # Perform K-means clustering
    attempts=10
    compactness,label,center=cv.kmeans(c_pixels,k,None,
                            criteria,attempts,
                            cv.KMEANS_PP_CENTERS)
    WSS.append(compactness)
    K.append(k)

K
WSS
```

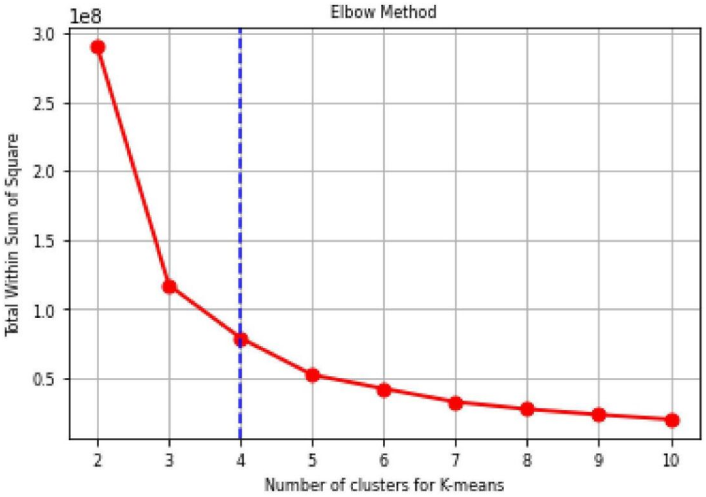
Out[10]:
```
[290110737.50625277,
 117299156.3951211,
 78932767.12288928,
 52262084.235240996,
 42166897.39009815,
 32576188.938685775,
 27308162.287027776,
 23311080.653457165,
 19860838.663348973]
```

In [11]:
```
np.round(np.array(WSS)/100000,2)
```

Out[11]:
```
array([2901.11, 1172.99,  789.33,  522.62,  421.67,  325.76,  273.08,
        233.11,  198.61])
```

In [12]:

```
plt.figure(figsize = (6, 4))
plt.plot(range(2, 11), np.round(WSS,3), ls='-',
         linewidth=2, color='red',marker='o',
         markerfacecolor='red', markersize=7)
plt.axvline(x=4, color='b',ls='--',linewidth=1.5)
#plt.axhline(y=78, color='b',ls='--',linewidth=2)
plt.title('Elbow Method',fontsize=8)
plt.xlabel('Number of clusters for K-means',fontsize=8)
plt.ylabel('Total Within Sum of Square',fontsize=8)
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.grid()
plt.savefig('D:/K-Means_Optimal number of Cluster for Image.jpg',dpi=700)
plt.show()
```



In [65]:

```
#==Optimal Value of k for Fuzzy C-means Clustering===#
K=4
```

In [66]:

```
attempts=10
compactness,label,center=cv.kmeans(c_pixels,K,None,
                                   criteria,attempts,
                                   cv.KMEANS_PP_CENTERS)
compactness
```

Out[66]:

```
78932776.78585124
```

In [67]:

```
label
```

Out[67]:

```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [0],
       [0]], dtype=int32)
```

In [68]:

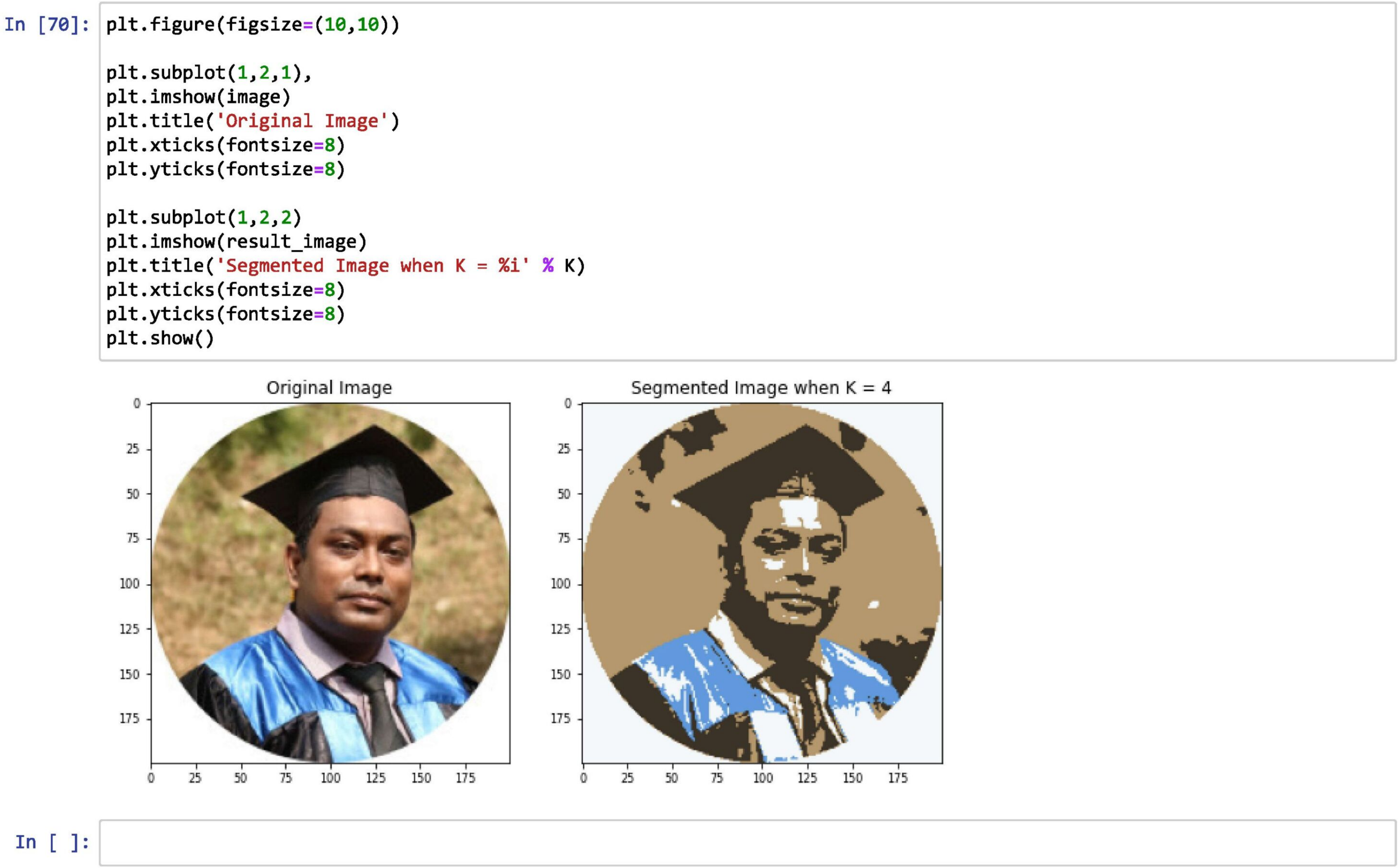
```
center = np.uint8(center)
center
```

Out[68]:

```
array([[243, 247, 248],
       [180, 151, 107],
       [ 58,  49,  39],
       [ 96, 154, 220]], dtype=uint8)
```

In [69]:

```
res = center[label.flatten()]
result_image = res.reshape((image.shape))
```



In [70]:

```
plt.figure(figsize=(10,10))

plt.subplot(1,2,1),
plt.imshow(image)
plt.title('Original Image')
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)

plt.subplot(1,2,2)
plt.imshow(result_image)
plt.title('Segmented Image when K = %i' % K)
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.show()
```

In [ ]:

====K-Means Clustering for Image====#