2. Problem Statement

In this assignment students have to compress racoon grey scale image into 5 clusters. In

the end, visualize both raw and compressed image and look for quality difference.

The raw image is available in spicy.misc package with the name face.

Hint:

import numpy as np

from sklearn import cluster, datasets

from scipy import misc

NOTE: The solution shared through Github should contain the source code used and

the screenshot of the output.

**Image compression using K-means clustering algorithms**

**Problem Statement**

In this assignment students have to compress racoon grey scale image into 5 clusters. In the end, visualize both raw and compressed image and look for quality difference.

The raw image is available in spicy.misc package with the name face.

Using K-means clustering, we will perform quantization of colours present in the image which will further help in compressing the image.

* In a coloured image, each pixel is of size 3 bytes (RGB), where each colour can have intensity values from 0 to 255.
* Following combinatorics, the total number of colours which can be represented are 256 *256*256.
* Practically, we are able to visualize only a few colours in an image.
* Shown below is an image of 1280 x 720 pixels taking 1.71 MB in PNG format.
* PNG is a lossless compression technique for images.
* **Our objective is to compress the image further using colour quantization, though the compression will be lossy.**

**In our problem of image compression, K-means clustering will group similar colours together in ‘k’ clusters (say ‘k’ = 128).**

* Therefore, the centroid of each cluster is representative of the 3 dimensional colour vectors (RGB) falling in the respective cluster.
* By now you might have understood what are we trying to do.
* These **‘k’ centroids will replace all the colour vectors in their clusters, thereby keeping only ‘k’ colour combinations for the whole image.**
* Thus, **we need to keep only the label of each pixel in the image that tells about the cluster in which that pixel falls.**
* **Also, we keep the ‘k’ centroids as codebook which are the only colours seen in the compressed image.**

**Compression**

* We will write a simple python code to compress the image and store the compressed image along with the code book.
* The compressed image saved here is nothing but the **cluster label of each pixel of the original image. Codebook is the fancy name given to the list of cluster centers (3-d RGB) achieved after running k-means algorithm.**
* Afterwards, both the arrays (the cluster labels and the codebook) are saved in data type ‘unsigned integer’ as the range of intensity values (0-255) and value of ‘k’ is always going to be less than 255 . The code given below does all this.

# importing important library

from skimage import io

from sklearn.cluster import KMeans

import numpy as np

from sklearn import cluster, datasets

from scipy import misc

import matplotlib.pyplot as plt

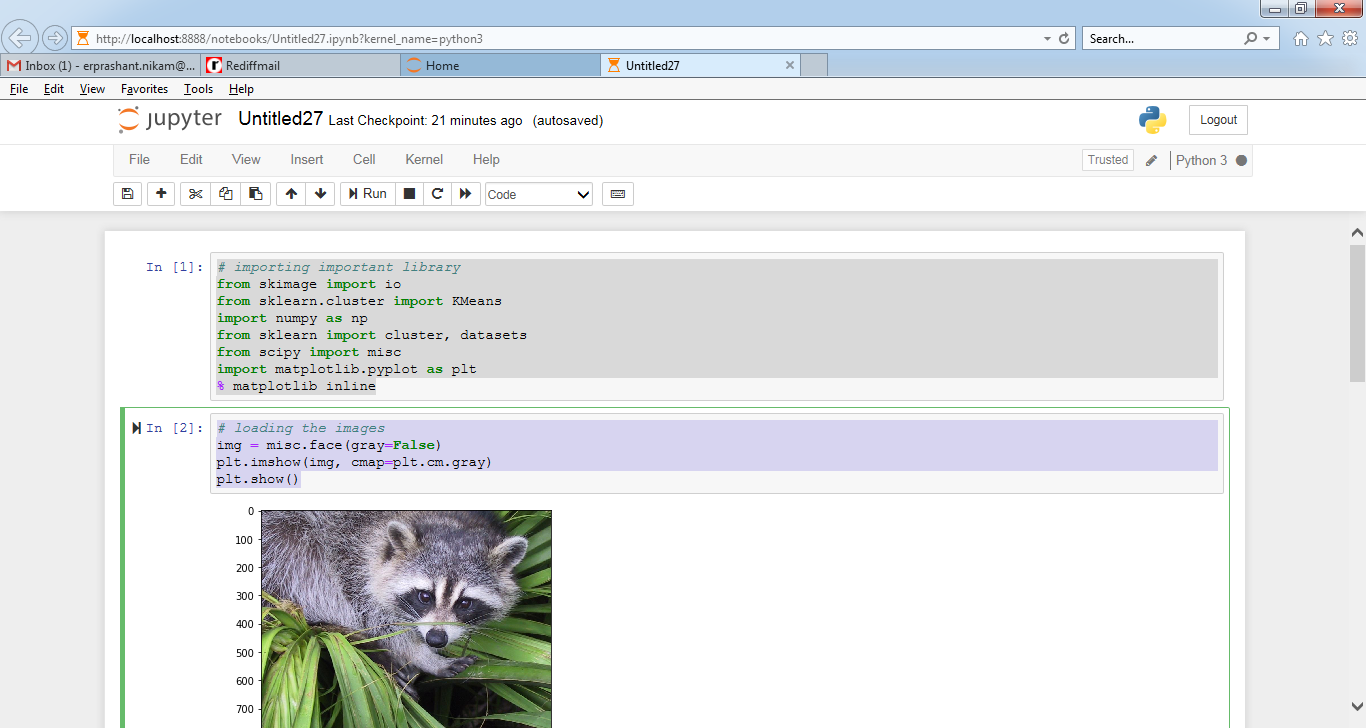
% matplotlib inline

# loading the images

img = misc.face(gray=False)

plt.imshow(img, cmap=plt.cm.gray)

plt.show()



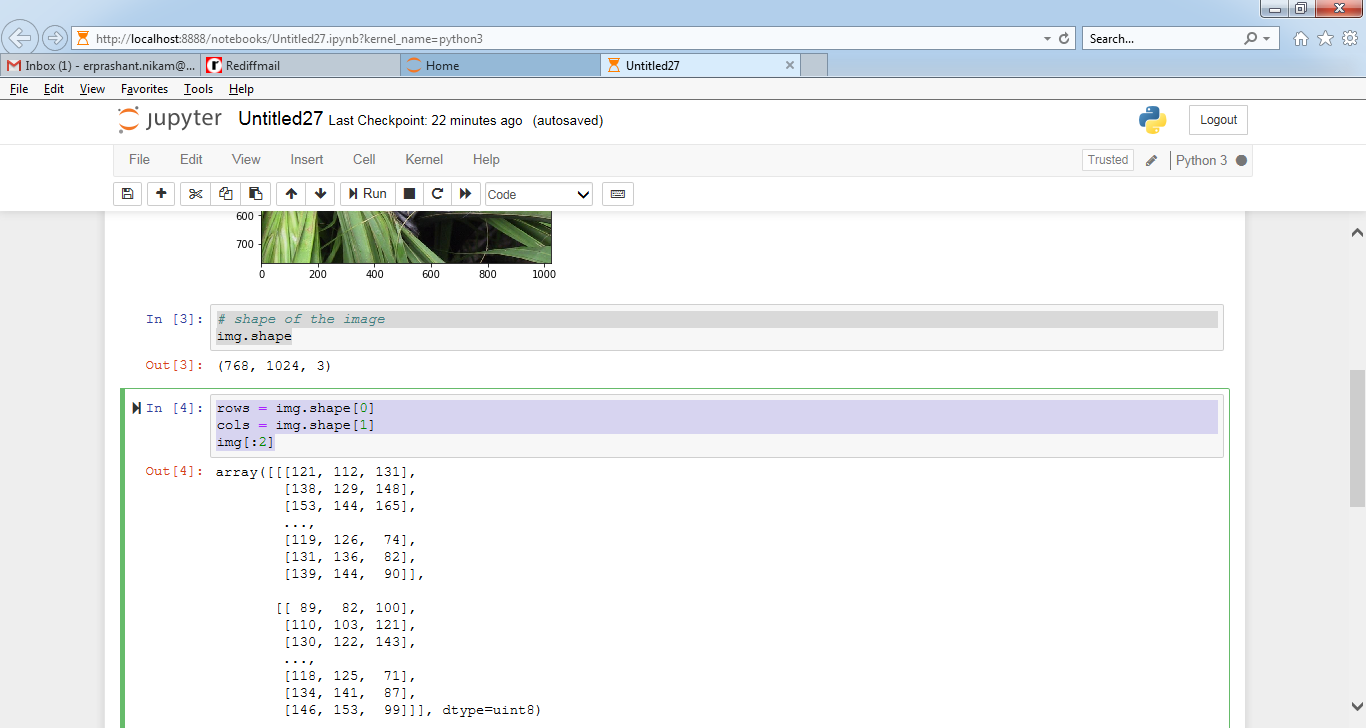
# shape of the image

img.shape

rows = img.shape[0]

cols = img.shape[1]

img[:2]



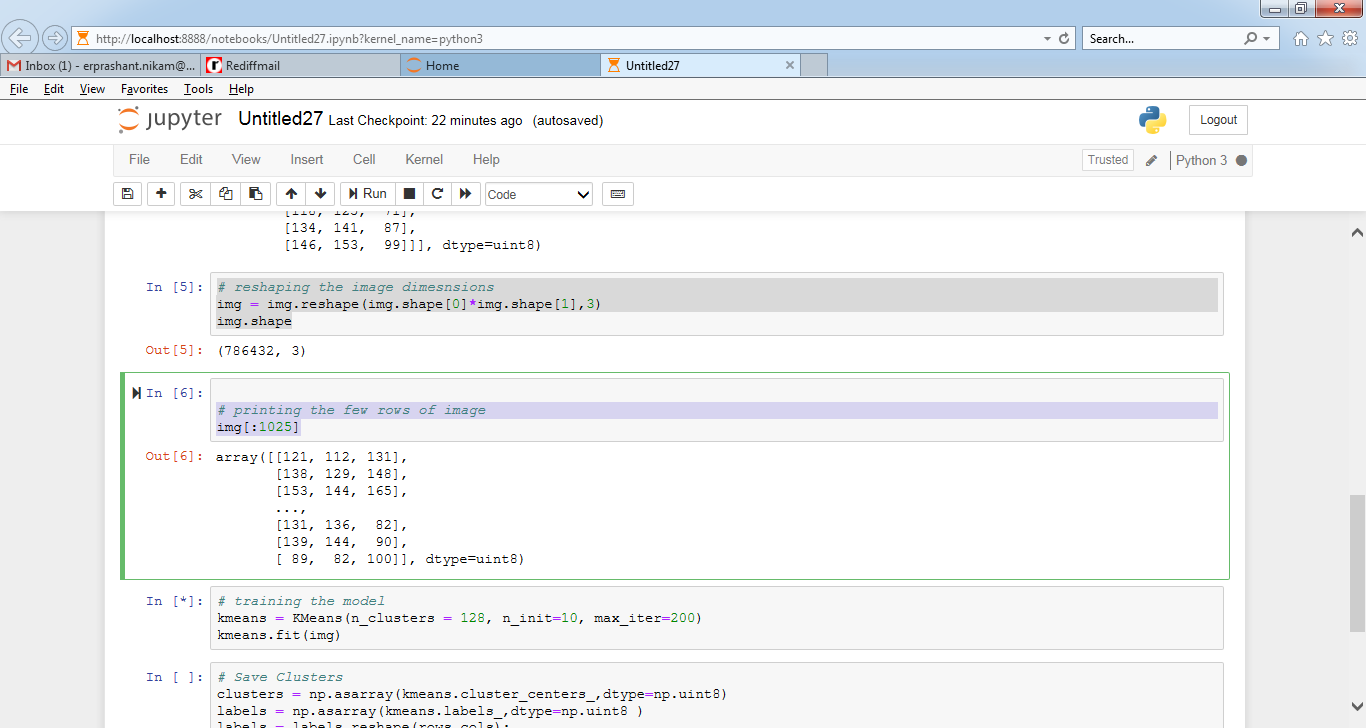
# reshaping the image dimesnsions

img = img.reshape(img.shape[0]\*img.shape[1],3)

img.shape

# printing the few rows of image

img[:1025]



# training the model

kmeans = KMeans(n\_clusters = 128, n\_init=10, max\_iter=200)

kmeans.fit(img)

# Save Clusters

clusters = np.asarray(kmeans.cluster\_centers\_,dtype=np.uint8)

labels = np.asarray(kmeans.labels\_,dtype=np.uint8 )

labels = labels.reshape(rows,cols);

np.save('codebook\_racoon.npy',clusters)

io.imsave('compressed\_racoon.png',labels)

**Decompression**

We also need to decompress the image in order to visualise the reconstructed image which is obviously an outcome of **lossy compression** performed.

* Below code does the decompression by assigning the 3-d colours from the code book to the each pixel depending upon its label.

centers = np.load('codebook\_racoon.npy')

c\_image = io.imread('compressed\_racoon.png')

image = np.zeros((c\_image.shape[0],c\_image.shape[1],3),dtype=np.uint8 )

**for** i **in** range(c\_image.shape[0]):

**for** j **in** range(c\_image.shape[1]):

image[i,j,:] = centers[c\_image[i,j],:]

io.imsave('reconstructed\_racoon.png',image);

io.imshow(image)

io.show()

**Compressed racoon image size: 462 KB**

**Decompressed racoon image size: 815 KB**

We can see the reconstructed image after decompression below. **Though the reconstructed image has lost a lot of pixel colour information but still you won’t find any major difference visually.**

Also, you can visualise these 128 colours found in the reconstructed image by viewing the colours in the codebook separately (may be by displaying **mono-coloured square box**). These colours are the centroids of clusters formed after performing k-means on original image.