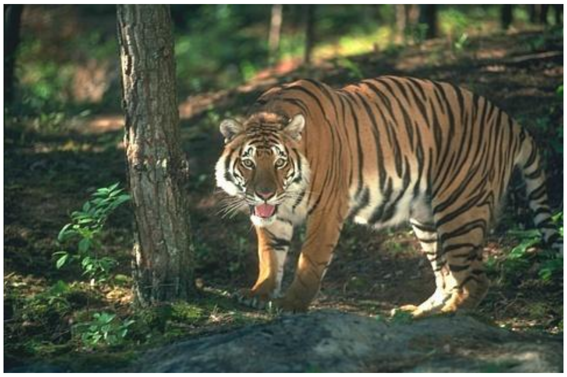
To solve this problem, GMM based clustering was used to segment a color image. The following image was chosen by me from the Berkeley Segmentation Dataset (`108005.jpg`):

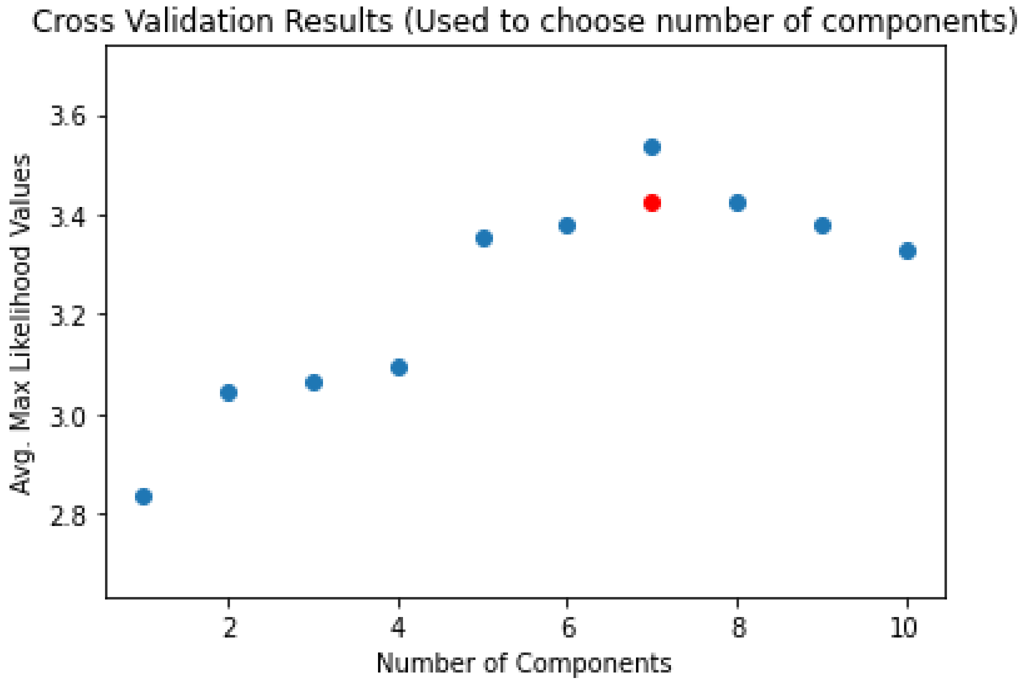


To proceed further with the solution, the following steps were followed as per the question:

1. Append row index, column index
2. Append red value, green value, blue value to a raw feature vector
3. Normalize each feature entry individually to the interval [0,1] so that all the feature vectors representing every pixel fit into the 5-D hypercube

After the above preprocessing is performed, we fit a Gaussian Mixture Model using the GaussianMixture library from sklearn using the Maximum Likelihood Parameter Estimation technique with K fold cross validation. The K (here, K=10) fold technique is used for determining the ideal number of Gaussian components needed for getting the best value of Max Likelihood Parameter Estimation (being used for Model Order Selection).

Following is the graph which shows the Average Max likelihood Parameter Estimation values v/s the number of components used in the training of the Gaussian Mixture:



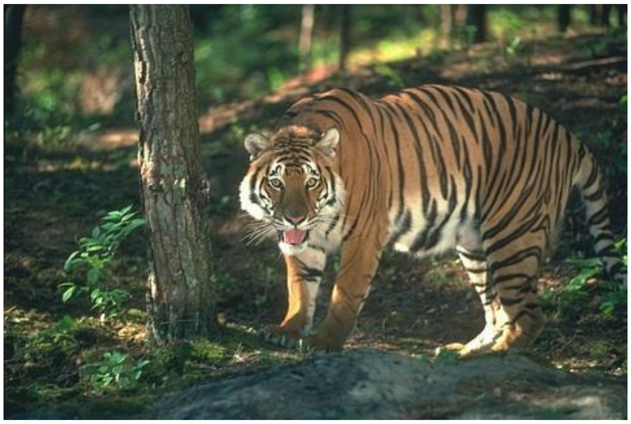
As the values are close to each other, following is a table which more precisely shows the values obtained:

Number of Components	Avg. Max Likelihood Estimation (MLE)
1	2.8355924279918736
2	3.04698074301517
3	3.0654636527081527
4	3.09506823571486
5	3.356142839313496
6	3.381608074140304
7	3.5379914612052445
8	3.423106441866486
9	3.3814435054710095
10	3.3290349741195415

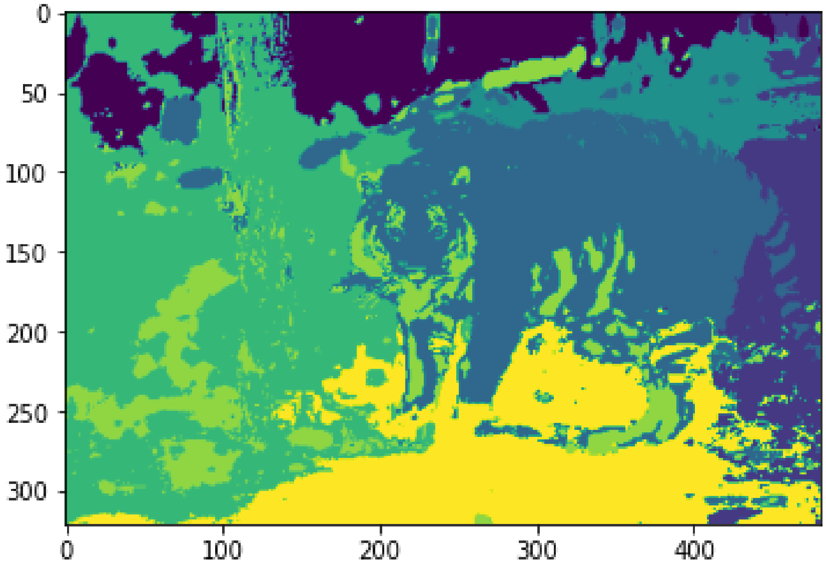
As we can see that the number of components to be chosen should be the index of the maximum value of the MLE, hence the ideal number of components chosen is **7** having the AVG MLE value of `3.5379914612052445`

After obtaining the value of number of components, the model was trained again with the Gaussian Mixture and by fitting the hypercube vector generated. This is being used to segment the image with the number of components equal to the number of components equal in the Gaussian Mixture Model.

The segmentation result of the image and the original image are shown below:



Original Image



Segmentation Result with Number of Components = 7

CODE STARTS HERE:

```
#Importing all the required libraries for GMM based clustering

import numpy as np
from sklearn.preprocessing import normalize
import cv2
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.mixture import GaussianMixture
from scipy.stats import multivariate_normal
import matplotlib.pyplot as plt


#------------------------------------------------------------------

#Importing the image and storing its shape (pixels) and mean value

image = cv2.imread("108005.jpg")
shape = image.shape
means = cv2.mean(image)

#Generating a 5 dimensional feature vector for image preprocessing (hypercube), and then normalizing each entry

hc = np.zeros((shape[0]*shape[1],5))
row = 0
for i in range(shape[0]):
  for j in range(shape[1]):
    hc[row,1] = j
    hc[row,0] = i
    for k in range(3):
      hc[row,k+2] = image[i,j,k]
    row=row+1
hc = normalize(hc,axis=0,norm='max')

#------------------------------------------------------------------

#Applying K fold cross validation and fitting a Gaussian Mixture Model to the features

cross_val_result = []
for m in range(1,11):
  ml = []
  kf = KFold(n_splits=10)
  for train_index, validation_index in kf.split(hc):
    gmm_fv = GaussianMixture(n_components=m,max_iter=1000).fit(hc[train_index,:])
    ml.append(gmm_fv.score(hc[validation_index,:]))
  cross_val_result.append(np.mean(ml))

#---------------------------------------------------------------------

#Deriving the model order selection and testing it on the 5-D hypercube

list_ = []
for i in range(len(cross_val_result)):
  list_.append(i)
components = list_[np.argmax(np.array(cross_val_result))] + 1

gmm_fv_test = GaussianMixture(n_components=components,max_iter=1000,init_params='kmeans').fit(hc)
final  = np.zeros((shape[0]*shape[1],components))

#---------------------------------------------------------------------

#Obtaining the array which contains the GMM based segmentation labels

for i in range(components):
  pdf = multivariate_normal.pdf(hc,mean=gmm_fv_test.means_[i,:],cov=gmm_fv_test.covariances_[i,:,:])
  final[:,i] = np.array(gmm_fv_test.weights_[i]*pdf)
final = np.argmax(final,axis=1)

#Displaying the GMM based labels for ideal number of components after performing Model Order Selection

plt.imshow(final.reshape(shape[0],shape[1]))
plt.show()


#---------------------------------------------------------------------

#Displaying graph which shows the Cross validation results, and also the number of components to be chosen for most accuracy

plt.scatter(range(1,11),cross_val_result)
plt.scatter(components,cross_val_result[components],marker='o',color='r')
plt.xlabel("Number of Components")
plt.ylabel("Avg. Max Likelihood Values")
plt.title("Cross Validation Results (Used to choose number of components)"
)
plt.ylim(min(cross_val_result)-0.2,max(cross_val_result)+0.2)
plt.show()


#---------------------------------------------------------------------

#Printing the cross validation results for easy visualization of values of
 the graph

print(cross_val_result)
```