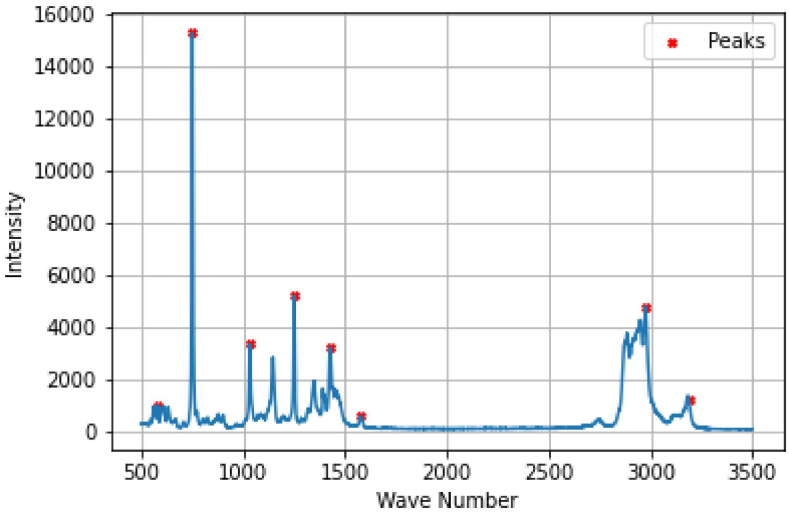
Name: Venkata Meghana Achanta USC ID: 2578990261

Q1a) To print the peaks with maximum intensity, I used the scipy find peaks function where I used the mean of the intensities as the height of the peak and with threshold as 6.4.

The following is the output obtained when I sorted the Peaks according to their intensities and displayed their corresponding wave number:

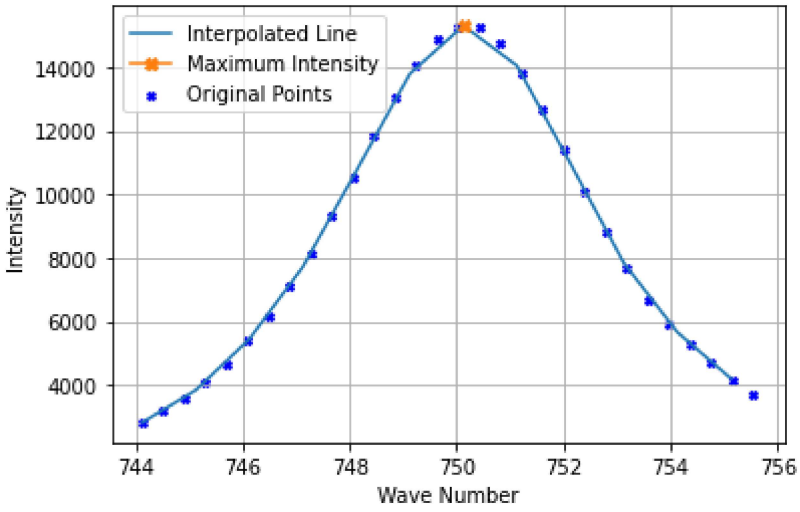
```
[750.42657, 1250.886, 2974.7341, 1031.9596, 1427.287, 3188.5417, 579.53802, 1575.338]
```

Q1b) The graph for the raw data of raman.txt plotted for Intensity vs Wave Number is as follows:

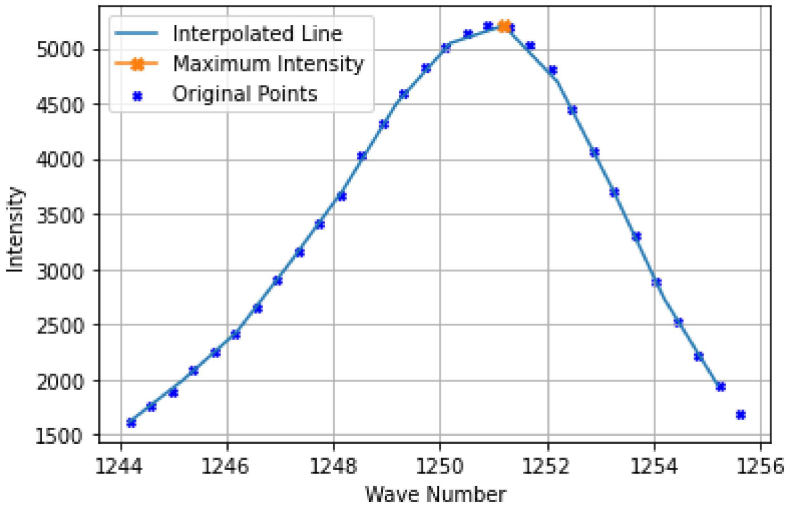


Q1c) To perform spline interpolation, I used the splrep and splev for performing interpolation and produced graphs for 4 zoomed in regions for regions of maximum intensity. I experimented with a few values and through observation, I assigned the smoothing factor as 0.25. The maximum intensity is obtained at the point where the slope is maximum.

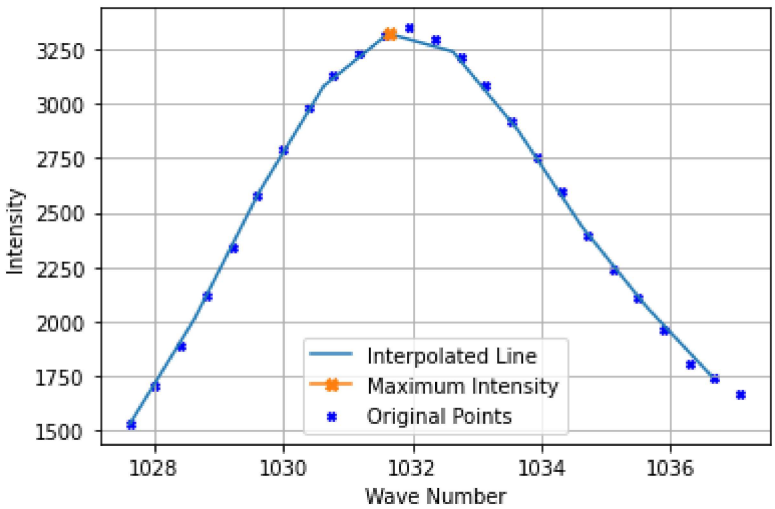
i. Region I



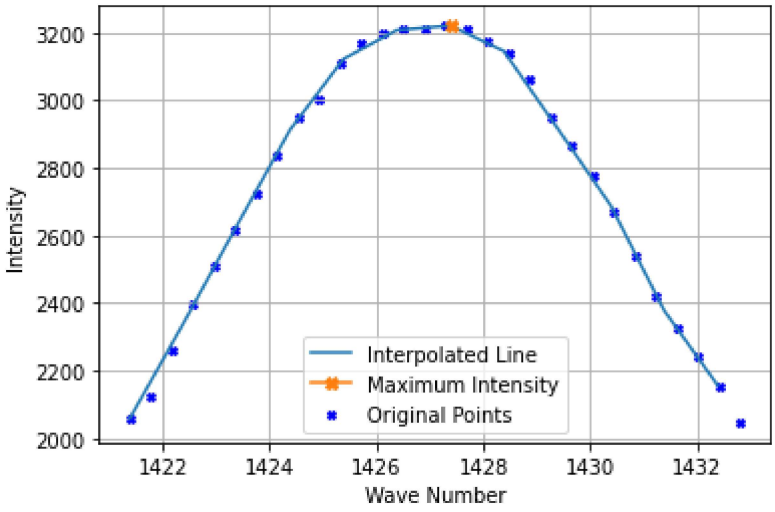
ii. Region II



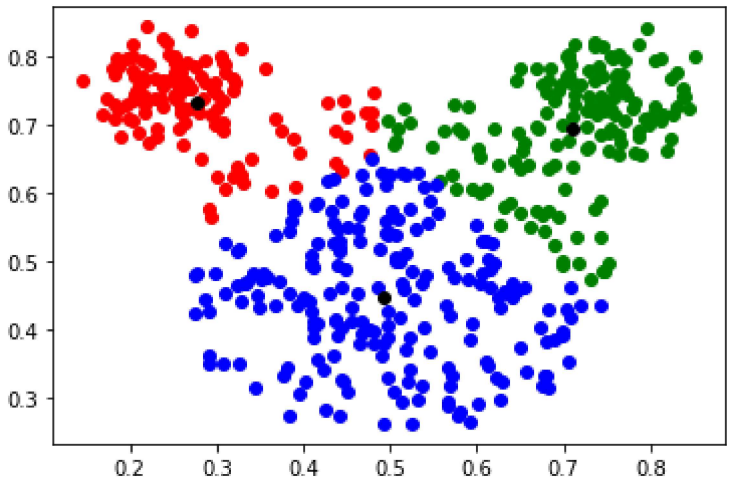
iii. Region III



iv. Region IV



Q2a) I used the scipy kmeans2 library to perform the kmeans clustering. I obtained the following plot after performing the clustering.

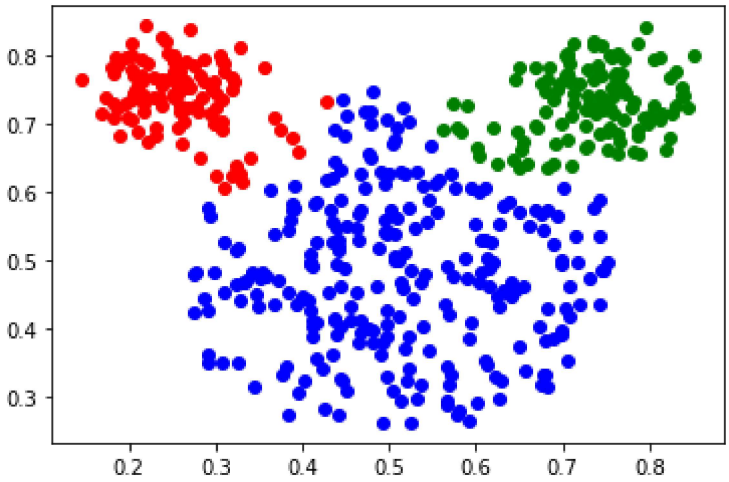


The obtained confusion matrix is as follows:

```
[[100   0   0]
 [  0 100   0]
 [ 25  54 211]]
```

Q2 b)

a) The model was run for 10 iterations after observing that the change in mean and covariance were very negligible after the 4th decimal point. The final scatter plot obtained after 10 iterations is as follows:

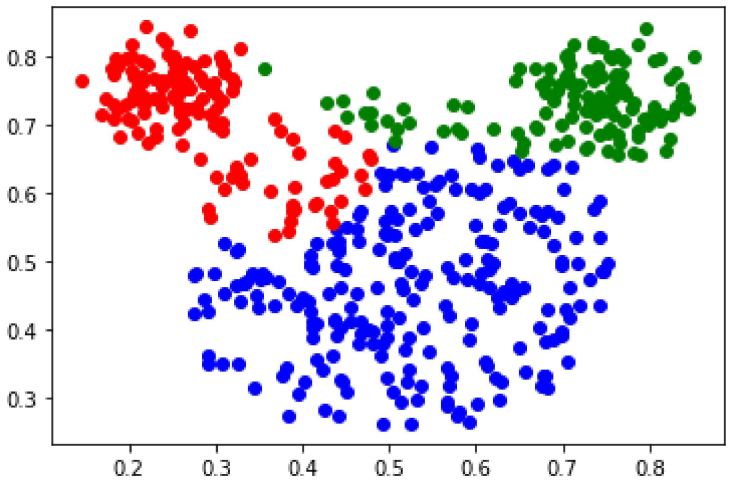


The confusion matrix obtained is as follows:

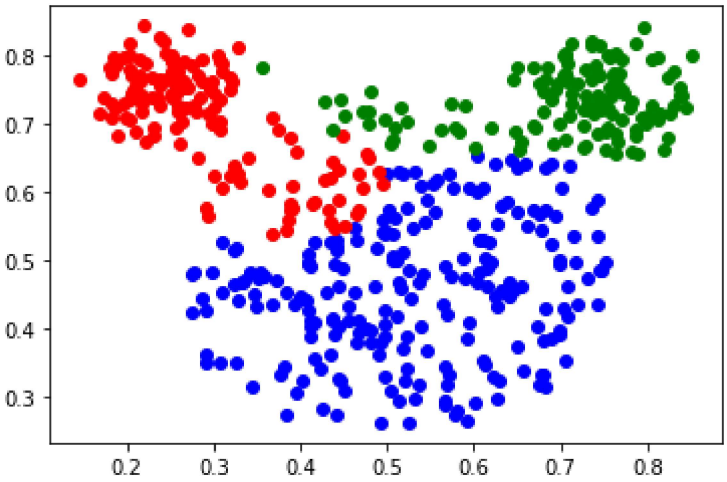
```
[[100   0   0]
 [  0 100   0]
 [ 12  18 260]]
```

b) The plots for the first four iterations are as follows:

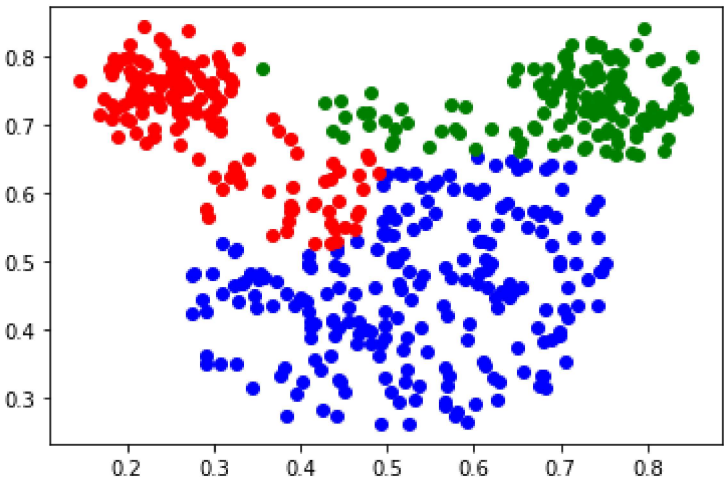
i.Iteration 1



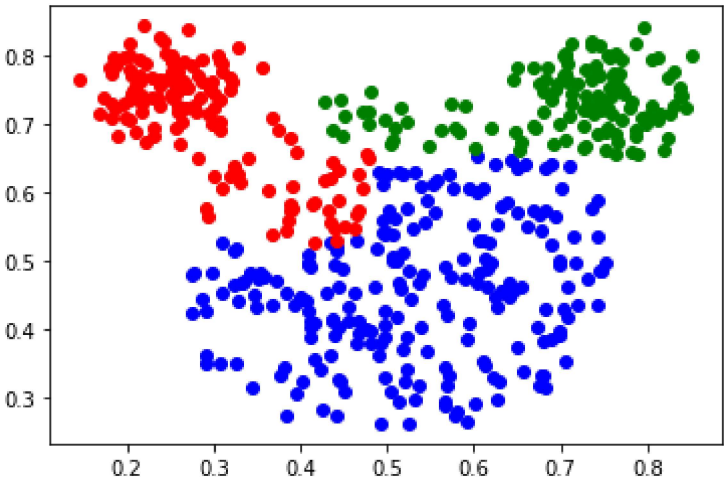
ii.Iteration 2



iii. Iteration 3



iv. Iteration 4



c) The scatterring plot obtained in part (a) wrongly predicted the values of the points which were labelled as 'Head'. As shown in the confusion matrix, out of the 290 points labelled as Head, only 211 points were correctly identified as Head whereas the rest 79 points were not predicted properly.

Using GMM, I observed that the model correctly predicted 260 points amongst 290 Head points as Heads. Only 30 points were incorrectly predicted.

Thus, GMM performs better as it uses both mean and covariance to assign clusters unlike kmeans which only uses Euclidean Distance as a measure to assign clusters. Furthermore, kmeans uses hard clustering and only uses spherical clusters whereas GMM uses elliptical clusters and performs soft clustering.

Appendix:

```
#Q1.
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
from scipy.signal import find_peaks
import sys
from scipy.interpolate import UnivariateSpline

#Loading the dataset raman.txt and converting it to a numpy array
dt = np.loadtxt("raman.txt", dtype=float)
x=[]
y=[]
for i in range(len(dt)):
  x.append(dt[i][0])
  y.append(dt[i][1])
x = np.array(x)
y = np.array(y)

#Raman Spectroscopy Peak Calculation for raw data
peaks1 = find_peaks(y, height = 580, threshold = 6.4, distance = 360)
heights = peaks1[1]['peak_heights']
pos = x[peaks1[0]]

#Sorting the peaks with the respective wave number
heights1, pos2 = (list(x) for x in zip(*sorted(zip(heights, pos), reverse = True)))
#Printing the wavenumbers corresponding to the 8 peaks
for i in range(8):
  sys.stdout.write(str(pos2[i]) + " ")

#Plotting the Peaks using find_peaks
fig = plt.figure()
ax = fig.subplots()
ax.plot(x,y)
ax.scatter(pos, heights, color = 'r', s = 15,marker = 'X', Label = 'Peaks')
plt.xlabel("Wave Number")
plt.ylabel("Intensity")
ax.legend()
```

```
ax.grid()
plt.show()

#Using arange function to get evenly spaced wavenumbers on the x-axis
x_fit = np.arange(x[620],x[650])

#Using splrep to define the splines
splines = interpolate.splrep(x,y, s = 0.25, k = 3)

#Using the splev to calculate the new intensity using the tuple returned by splrep and the x_
y2 = interpolate.splev(x_fit, splines)

x1 = []
y1 = []
for i in range(620,650):
  x1.append(x[i])
  y1.append(y[i])
x1 = np.array(x1)
y1 = np.array(y1)

peaks = find_peaks(y2, height = 100, threshold = 6.4, distance = 100)
heights1 = peaks[1]['peak_heights'][0]
pos1 = x_fit[peaks[0][0]]


fig = plt.figure()
ax = fig.subplots()
ax.plot(x_fit,y2,Label = 'Interpolated Line')
ax.plot(pos1,heights1, Label = 'Maximum Intensity', Marker = 'X')

ax.scatter(x1, y1, color = 'blue', s = 15,marker = 'X', Label = 'Original Points')
plt.xlabel("Wave Number")
plt.ylabel("Intensity")


ax.legend()
ax.grid()

plt.show()

#Using arange function to get evenly spaced wavenumbers on the x-axis
x_fit = np.arange(x[1890],x[1920])

#Using splrep to define the splines
splines = interpolate.splrep(x,y, s = 0.25, k = 3)

#Using the splev to calculate the new intensity using the tuple returned by splrep and the x_
y2 = interpolate.splev(x_fit, splines)

x1 = []
y1 = []
```

```
for i in range(1890,1920):
  x1.append(x[i])
  y1.append(y[i])
x1 = np.array(x1)
y1 = np.array(y1)

peaks = find_peaks(y2, height = 100, threshold = 6.4, distance = 100)
heights1 = peaks[1]['peak_heights'][0]
pos1 = x_fit[peaks[0][0]]


fig = plt.figure()
ax = fig.subplots()
ax.plot(x_fit,y2,Label = 'Interpolated Line')
ax.plot(pos1,heights1, Label = 'Maximum Intensity', Marker = 'X')

ax.scatter(x1, y1, color = 'blue', s = 15,marker = 'X', Label = 'Original Points')

plt.xlabel("Wave Number")
plt.ylabel("Intensity")
ax.legend()
ax.grid()

plt.show()

#Using arange function to get evenly spaced wavenumbers on the x-axis
x_fit = np.arange(x[1340],x[1365])

#Using splrep to define the splines
splines = interpolate.splrep(x,y, s = 0.25, k = 3)

#Using the splev to calculate the new intensity using the tuple returned by splrep and the x_
y2 = interpolate.splev(x_fit, splines)

x1 = []
y1 = []
for i in range(1340,1365):
  x1.append(x[i])
  y1.append(y[i])
x1 = np.array(x1)
y1 = np.array(y1)

peaks = find_peaks(y2, height = 558, threshold = 6.4, distance = 100)
heights1 = peaks[1]['peak_heights'][0]
pos1 = x_fit[peaks[0][0]]


fig = plt.figure()
ax = fig.subplots()
ax.plot(x_fit,y2,Label = 'Interpolated Line')
ax.plot(pos1,heights1, Label = 'Maximum Intensity', Marker = 'X')
```

```
ax.scatter(x1, y1, color = 'blue', s = 15,marker = 'X', Label = 'Original Points')

plt.xlabel("Wave Number")
plt.ylabel("Intensity")
ax.legend()
ax.grid()

plt.show()

#Using arange function to get evenly spaced wavenumbers on the x-axis
x_fit = np.arange(x[2340],x[2370])

#Using splrep to define the splines
splines = interpolate.splrep(x,y, s = 0.25, k = 3)

#Using the splev to calculate the new intensity using the tuple returned by splrep and the x_
y2 = interpolate.splev(x_fit, splines)

x1 = []
y1 = []
for i in range(2340,2370):
  x1.append(x[i])
  y1.append(y[i])
x1 = np.array(x1)
y1 = np.array(y1)

peaks = find_peaks(y2, height = 558, threshold = 6.4, distance = 100)
heights1 = peaks[1]['peak_heights'][0]
pos1 = x_fit[peaks[0][0]]


fig = plt.figure()
ax = fig.subplots()
ax.plot(x_fit,y2,Label = 'Interpolated Line')
ax.plot(pos1,heights1, Label = 'Maximum Intensity', Marker = 'X')

ax.scatter(x1, y1, color = 'blue', s = 15,marker = 'X', Label = 'Original Points')

plt.xlabel("Wave Number")
plt.ylabel("Intensity")
ax.legend()
ax.grid()

plt.show()



#Q2a

import numpy as np
```

```
from numpy import loadtxt
import matplotlib.pyplot as plt
from scipy.cluster.vq import kmeans2

values = np.loadtxt('cluster.txt', comments='#' ,delimiter=' ', usecols=[0,1])
labels = np.loadtxt('cluster.txt', comments = '#' ,delimiter=' ', usecols=[2], dtype = 'str')

cluster_1 = values[labels == 'Head']
cluster_2 = values[labels == 'Ear_right']
cluster_3 = values[labels == 'Ear_left']
plt.scatter(cluster_1[:,0], cluster_1[:,1], label ='Head', color = 'blue')
plt.scatter(cluster_2[:,0], cluster_2[:,1], label ='Ear_Right', color = 'green')
plt.scatter(cluster_3[:,0], cluster_3[:,1], label ='Ear_Left', color = 'red')
plt.show()

centroids, labels2 = kmeans2(values, 3, minit = 'points')

cluster_1 = values[labels2 == 0]
cluster_2 = values[labels2 == 1]
cluster_3 = values[labels2 == 2]

l = []
for i in labels2:
  if(i==2):
    l.append('Head')
  elif(i==1):
    l.append('Ear_left')
  else:
    l.append('Ear_right')

l = np.array(l)

plt.scatter(cluster_1[:,0], cluster_1[:,1], label ='Head', color = 'Blue')
plt.scatter(cluster_2[:,0], cluster_2[:,1], label ='Ear_right', color = 'red')
plt.scatter(cluster_3[:,0], cluster_3[:,1], label ='Ear_left', color = 'green')
plt.scatter(centroids[:,0], centroids[:,1], label = 'Centroids', color = 'black')
plt.show()

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(labels, l)
print(cm)



#Q2b Gaussian Mixture Models

import numpy as np
from numpy import loadtxt
from scipy.stats import multivariate_normal
import matplotlib.pyplot as plt
from scipy.cluster.vq import kmeans2

values = np.loadtxt('cluster.txt', comments='#' ,delimiter=' ', usecols=[0,1])
```

```
values = np.loadtxt('cluster.txt', comments = '#' ,delimiter=' ', usecols=[0,1])
labels = np.loadtxt('cluster.txt', comments = '#' ,delimiter=' ', usecols=[2], dtype = 'str')
print(values.shape)

labels = np.array(labels)

centroids, labels2 = kmeans2(values, 3, minit = 'points')
print(centroids)

l1 = []
l2 = []
l3 = []
l = []
for i in labels2:
  if(i==0):
    l.append('Head')
  elif(i==2):
    l.append('Ear_left')
  else:
    l.append('Ear_right')

l = np.array(l)

for i in l:
  if (i=='Head'):
    l1.append([1,0,0])
  elif (i=='Ear_left'):
    l2.append([0,1,0])
  else:
    l3.append([0,0,1])
l1 = np.array(l1)
l2 = np.array(l2)
l3 = np.array(l3)
#print(l1.shape, l2.shape,l3.shape)
r = np.concatenate((l1,l2,l3), axis = 0)
print(r.shape)

gamma = r.copy()

mean = []
cov = []
wk = []


for i in range(3):
  mean.append((np.matmul(r[:,i].reshape(1,-1),values))/(np.sum(r[:,i])))
mean_np = np.concatenate(mean, axis = 0)

for i in range(3):
  sum1 = np.zeros((2,2))
  for j in range(490):
    sum1 = sum1 + r[j,i]*np.matmul((values[j] - mean_np[i]).reshape(-1,1),(values[j] - mean_n
```

```
    cov.append(sum1.reshape(1,2,2)/np.sum(r[:,i]))
  cov_np = np.concatenate(cov, axis=0)

  for i in range(3):
    wk.append(np.sum(r[:,i])/490)
  wk_np = np.array(wk)

  print(cov_np, mean_np)

  def pdf(x, mu, cov):
    # print(x, mu, cov)
    dist =  multivariate_normal(mu, cov)
    return dist.pdf(x)

  # EM Algorithm
  mean_iter = []
  cov_iter = []
  wk_iter = []
  iters = 10

  for i in range(iters):
    print("Iteration : ", i)
    #E Step
    r_temp = np.zeros((490,3))
    for i in range(3):
      for j in range(490):
        # print(pdf(values[j], mean_np[i], cov_np[i]))
        r_temp[j, i] = wk_np[i]*pdf(values[j], mean_np[i], cov_np[i])
    r_temp = r_temp/(np.sum(r_temp, axis = 1)).reshape(-1,1)
    r = r_temp.copy()

    #M Step
    mean = []
    cov = []
    wk = []


    for i in range(3):
      mean.append((np.matmul(r[:,i].reshape(1,-1),values))/(np.sum(r[:,i])))
    mean_np = np.concatenate(mean, axis = 0)
    mean_iter.append(mean_np)
    #print("Mean : ", mean_np)

    for i in range(3):
      sum1 = np.zeros((2,2))
      for j in range(490):
        sum1 = sum1 + r[j,i]*np.matmul((values[j] - mean_np[i]).reshape(-1,1),(values[j] - mean
      cov.append(sum1.reshape(1,2,2)/np.sum(r[:,i]))
    cov_np = np.concatenate(cov, axis=0)
    cov_iter.append(cov_np)
    #print("Covariance : ", cov_np)
```

```
    for i in range(3):
      wk.append(np.sum(r[:,i])/490)
    wk_np = np.array(wk)
    wk_iter.append(wk_np)
    #print("Weight : ", wk_np)

  def inf(X,mu,cov,w):
    scores = []
    for i in range(3):
      scores.append(w[i]*pdf(X,mu[i],cov[i]))
    return scores.index(max(scores))

  ref = []
  for i in range(490):
    ref.append(inf(values[i],mean_iter[0],cov_iter[0],wk_iter[0]))
  #print(ref)

  cluster_1 = values[np.array(ref) == 0]
  cluster_2 = values[np.array(ref) == 1]
  cluster_3 = values[np.array(ref) == 2]

  l = []
  for i in ref:
    if(i==0):
      l.append('Head')
    elif(i==1):
      l.append('Ear_left')
    else:
      l.append('Ear_right')

  l = np.array(l)

  from sklearn.metrics import confusion_matrix
  #print(labels)
  cm = confusion_matrix(labels, l)
  print(cm)
  #print(cluster_1)
  plt.scatter(cluster_1[:,0], cluster_1[:,1], label ='Head', color = 'Blue')
  plt.scatter(cluster_2[:,0], cluster_2[:,1], label ='Ear_right', color = 'red')
  plt.scatter(cluster_3[:,0], cluster_3[:,1], label ='Ear_left', color = 'green')
  #plt.scatter(centroids[:,0], centroids[:,1], label = 'Centroids', color = 'black')
  plt.show()

  ref1 = []
  for i in range(490):
    ref1.append(inf(values[i],mean_iter[1],cov_iter[1],wk_iter[1]))
  #print(ref)

  cluster_1 = values[np.array(ref1) == 0]
  cluster_2 = values[np.array(ref1) == 1]
  cluster_3 = values[np.array(ref1) == 2]
```

```
l = []
for i in ref1:
  if(i==0):
    l.append('Head')
  elif(i==1):
    l.append('Ear_left')
  else:
    l.append('Ear_right')

l = np.array(l)

from sklearn.metrics import confusion_matrix
#print(labels)
cm = confusion_matrix(labels, l)
print(cm)

plt.scatter(cluster_1[:,0], cluster_1[:,1], label ='Head', color = 'Blue')
plt.scatter(cluster_2[:,0], cluster_2[:,1], label ='Ear_right', color = 'red')
plt.scatter(cluster_3[:,0], cluster_3[:,1], label ='Ear_left', color = 'green')
#plt.scatter(centroids[:,0], centroids[:,1], label = 'Centroids', color = 'black')
plt.show()

ref2 = []
for i in range(490):
  ref2.append(inf(values[i],mean_iter[2],cov_iter[2],wk_iter[2]))
#print(ref2)

cluster_1 = values[np.array(ref2) == 0]
cluster_2 = values[np.array(ref2) == 1]
cluster_3 = values[np.array(ref2) == 2]

l = []
for i in ref2:
  if(i==0):
    l.append('Head')
  elif(i==1):
    l.append('Ear_left')
  else:
    l.append('Ear_right')

l = np.array(l)

from sklearn.metrics import confusion_matrix
#print(labels)
cm = confusion_matrix(labels, l)
print(cm)

plt.scatter(cluster_1[:,0], cluster_1[:,1], label ='Head', color = 'Blue')
plt.scatter(cluster_2[:,0], cluster_2[:,1], label ='Ear_right', color = 'red')
plt.scatter(cluster_3[:,0], cluster_3[:,1], label ='Ear_left', color = 'green')
#plt.scatter(centroids[:,0], centroids[:,1], label = 'Centroids', color = 'black')
```

```
plt.show()

ref3 = []
for i in range(490):
  ref3.append(inf(values[i],mean_iter[3],cov_iter[3],wk_iter[3]))
#print(ref2)

cluster_1 = values[np.array(ref3) == 0]
cluster_2 = values[np.array(ref3) == 1]
cluster_3 = values[np.array(ref3) == 2]

l = []
for i in ref3:
  if(i==0):
    l.append('Head')
  elif(i==1):
    l.append('Ear_left')
  else:
    l.append('Ear_right')

l = np.array(l)

from sklearn.metrics import confusion_matrix
#print(labels)
cm = confusion_matrix(labels, l)
print(cm)

plt.scatter(cluster_1[:,0], cluster_1[:,1], label ='Head', color = 'Blue')
plt.scatter(cluster_2[:,0], cluster_2[:,1], label ='Ear_right', color = 'red')
plt.scatter(cluster_3[:,0], cluster_3[:,1], label ='Ear_left', color = 'green')
plt.show()

ref10 = []
for i in range(490):
  ref10.append(inf(values[i],mean_iter[9],cov_iter[9],wk_iter[9]))
#print(ref2)

cluster_1 = values[np.array(ref10) == 0]
cluster_2 = values[np.array(ref10) == 1]
cluster_3 = values[np.array(ref10) == 2]

l = []
for i in ref10:
  if(i==0):
    l.append('Head')
  elif(i==1):
    l.append('Ear_left')
  else:
    l.append('Ear_right')

l = np.array(l)
```

```
from sklearn.metrics import confusion_matrix
#print(labels)
cm = confusion_matrix(labels, l)
print(cm)

plt.scatter(cluster_1[:,0], cluster_1[:,1], label ='Head', color = 'Blue')
plt.scatter(cluster_2[:,0], cluster_2[:,1], label ='Ear_right', color = 'red')
plt.scatter(cluster_3[:,0], cluster_3[:,1], label ='Ear_left', color = 'green')
#plt.scatter(centroids[:,0], centroids[:,1], label = 'Centroids', color = 'black')
plt.show()
```