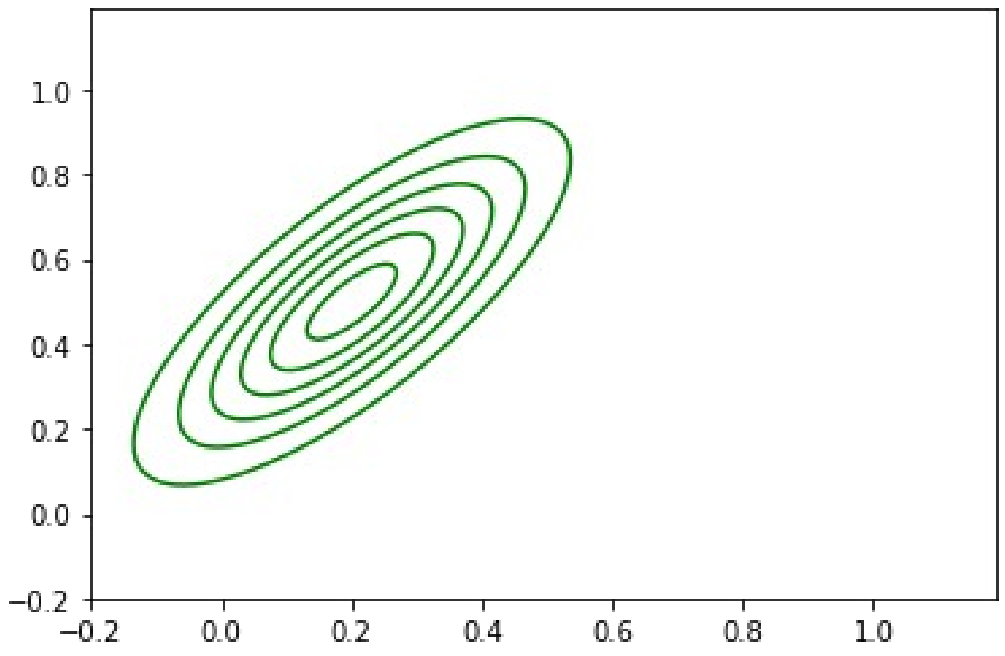
3 (Soft-assignment intuition) The objective of this exercise is to understand the multivariate gaussian distribution.

(a) Refer to figure 2.23 in Bishop and generate the plots similar to that. As a self study, explore how covariance matrix changes the contours.

```
# Importing required libraries
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal

# Generates numbers between 0, 1
x_vals = np.arange(-0.2, 1.2, 0.01)
```

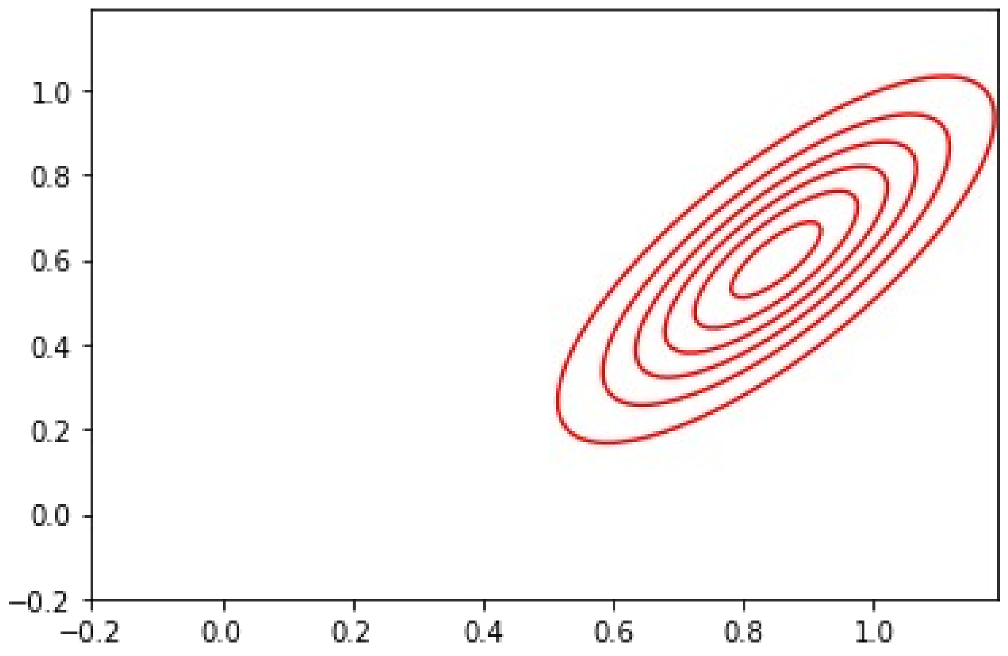
Plotting multivariate Gaussian distribution 1

```
# Assuming mean and covariance matrix suitably.
mean1 = [0.2, 0.5]
cov1 = [[0.03, 0.04], [0.03, 0.05]]
x1, y1 = np.meshgrid(x_vals, x_vals)
distr = multivariate_normal(mean1, cov1)

# Generating the density function
# for each point in the meshgrid

pos1 = np.zeros(x1.shape)
for i in range(x1.shape[0]):
    for j in range(x1.shape[1]):
        pos1[i,j] = distr.pdf([x1[i,j], y1[i,j]])

#p = multivariate_normal(x_vals, len(x_vals), mean, cov)
#print(p)

#print(x)
#print(len(x[0]))
#print(len(z))
#print(z)

#print(pos1)
plt.contour(x1, y1, pos1, colors='g')
plt.show()
```



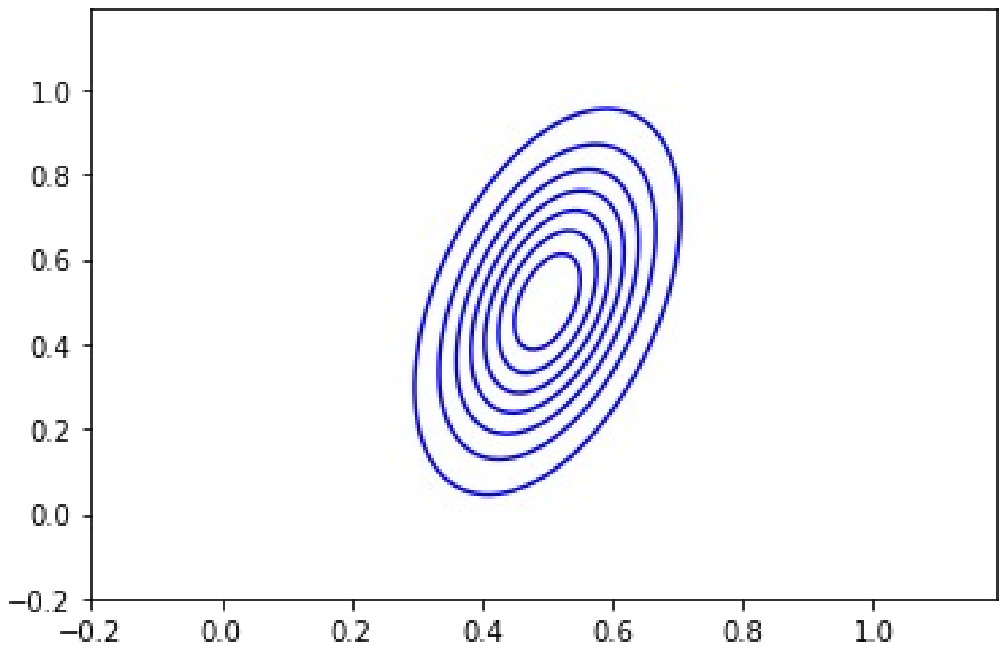
Plotting multivariate Gaussian distribution 2

```
# Assuming mean and covariance matrix suitably.
mean2 = [0.85, 0.6]
cov2 = [[0.03, 0.04], [0.03, 0.05]]
x2, y2 = np.meshgrid(x_vals, x_vals)
distr = multivariate_normal(mean2, cov2)

# Generating the density function
# for each point in the meshgrid

pos2 = np.zeros(x2.shape)
for i in range(x2.shape[0]):
    for j in range(x2.shape[1]):
        pos2[i,j] = distr.pdf([x2[i,j], y2[i,j]])

#p = multivariate_normal(x_vals, len(x_vals), mean, cov)
#print(p)

#print(x)
#print(len(x[0]))
#print(len(z))
#print(z)
plt.contour(x2, y2, pos2, colors='r')
plt.show()
```



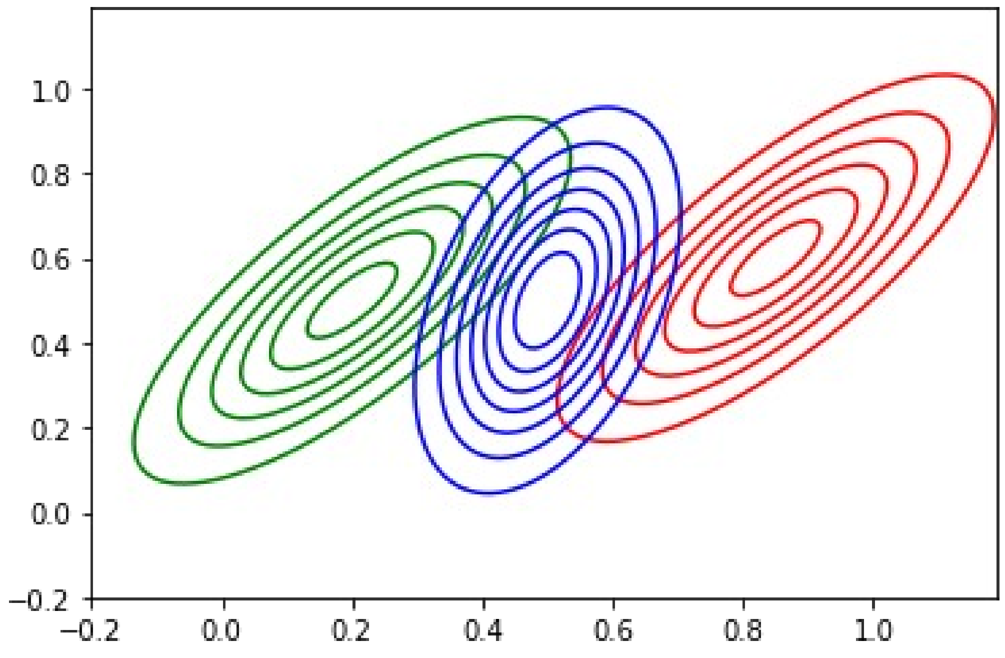
Plotting multivariate Gaussian distribution 3

```
# Assuming mean and covariance matrix suitably.
mean3 = [0.5, 0.5]
cov3 = [[0.01, 0.1], [0.01, 0.05]]
x3, y3 = np.meshgrid(x_vals, x_vals)
distr = multivariate_normal(mean3, cov3)


#Generating the density function for each point in the meshgrid

pos3 = np.zeros(x3.shape)
for i in range(x3.shape[0]):
    for j in range(x3.shape[1]):
        pos3[i,j] = distr.pdf([x3[i,j], y3[i,j]])

#p = multivariate_normal(x_vals, len(x_vals), mean, cov)
#print(p)

#print(x)
#print(len(x[0]))
#print(len(z))
#print(z)
plt.contour(x3, y3, pos3, colors='b')
plt.show()
```



Contour Plot of multivariate Gaussian distribution 1, 2, 3

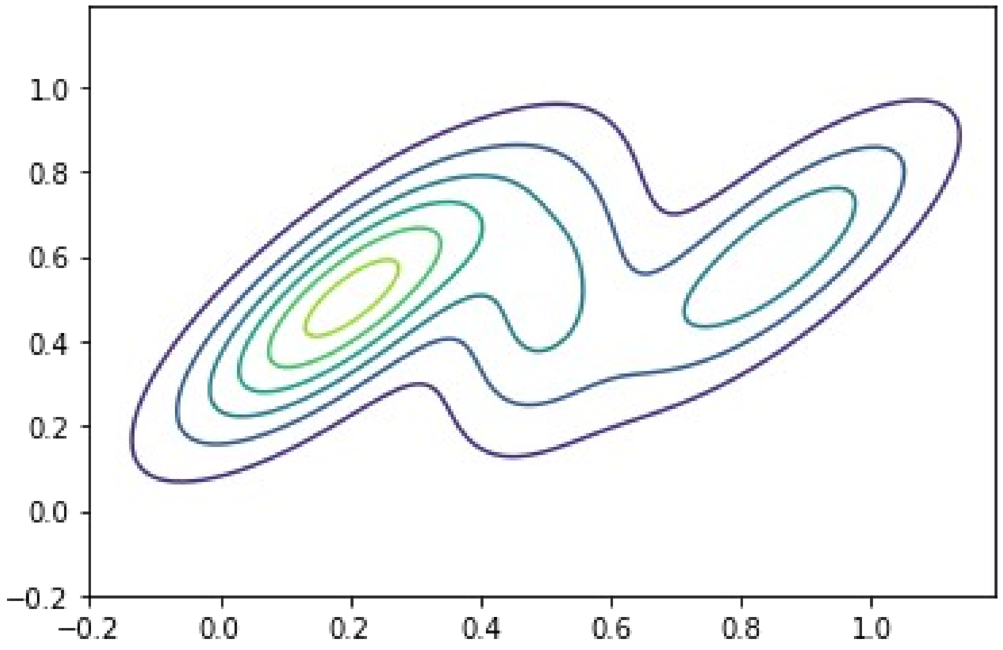
```
# Plotting contour of all three gaussian distribution.
plt.contour(x1, y1, pos1, colors='g')
plt.contour(x2, y2, pos2, colors='r')
plt.contour(x3, y3, pos3, colors='b')

plt.show()
```



OBSERVATIONS :-

In above fig, Contours of constant density for each of the mixture components, in which the 3 components are denoted red, blue and green, and the values of the mixing coefficients are shown below each component.

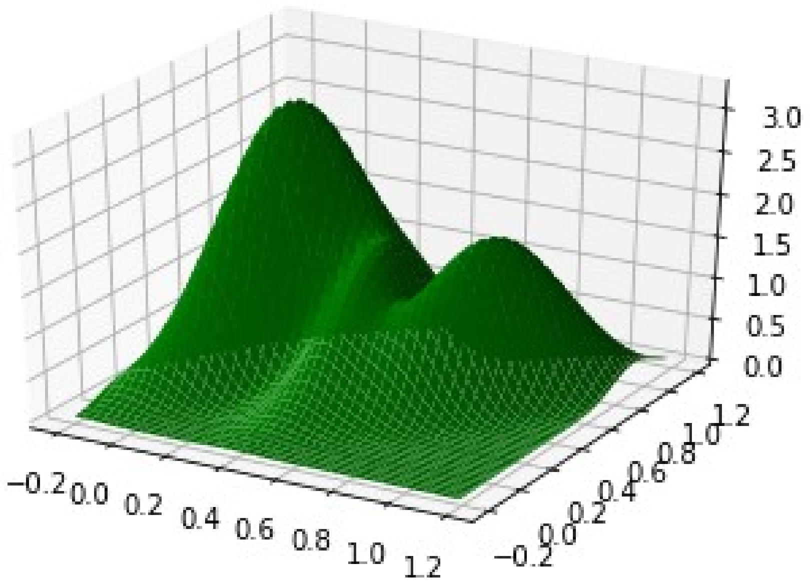
```
# Fixing mixing coefficients of all three gaussian distribution
pi1 = 0.5
pi2 = 0.3
pi3 = 0.2

x, y = np.meshgrid(x_vals, x_vals)

# Calculating Pdf of the mixture.
pos = pi1*pos1 + pi2*pos2 + pi3*pos3

#Plotting contour of the mixture.
plt.contour(x, y, pos)
plt.show()
```



In above fig, Contours of the marginal probability density P(x) of the mixture distribution.

```
# Fixing mixing coefficients of all three gaussian distribution
pi1 = 0.5
pi2 = 0.3
pi3 = 0.2

x, y = np.meshgrid(x_vals, x_vals)

# Calculating Pdf of the mixture.
```

```
pos = pi1*pos1 + pi2*pos2 + pi3*pos3

# Surface plot of a mixture.
ax = plt.axes(projection='3d')
ax.plot_surface(x, y, pos, color="green")
#ax.plot_wireframe(x, y, pos, color='red')
#ax.set_title('wireframe')
plt.show()
```



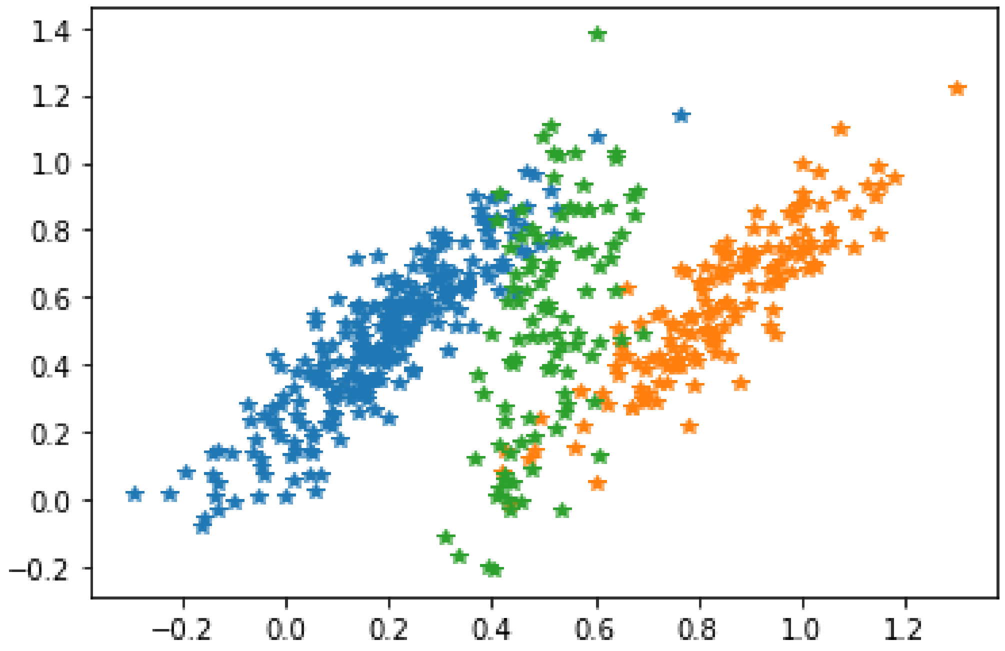
```
In above fig, A surface plot of the distribution P(x)
```

3(b) Generate 500 points from previous step clearing indicating the responsible cluster. Thereafter, generate figure 9.5 from Bishop.

```
# Taking 50% points from Gaussian Distribution 1
x1, y1 = np.random.multivariate_normal(mean1, cov1, 250).T

# Taking 30% points from Gaussian Distribution 2
x2, y2 = np.random.multivariate_normal(mean2, cov2, 150).T

# Taking 20% points from Gaussian Distribution 3
x3, y3 = np.random.multivariate_normal(mean3, cov3, 100).T

# plotting the three Gaussian distribution as a mixture.
plt.plot(x1, y1, '*')
plt.plot(x2, y2, '*')
plt.plot(x3, y3, '*')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2:
RuntimeWarning: covariance is not positive-semidefinite.
```

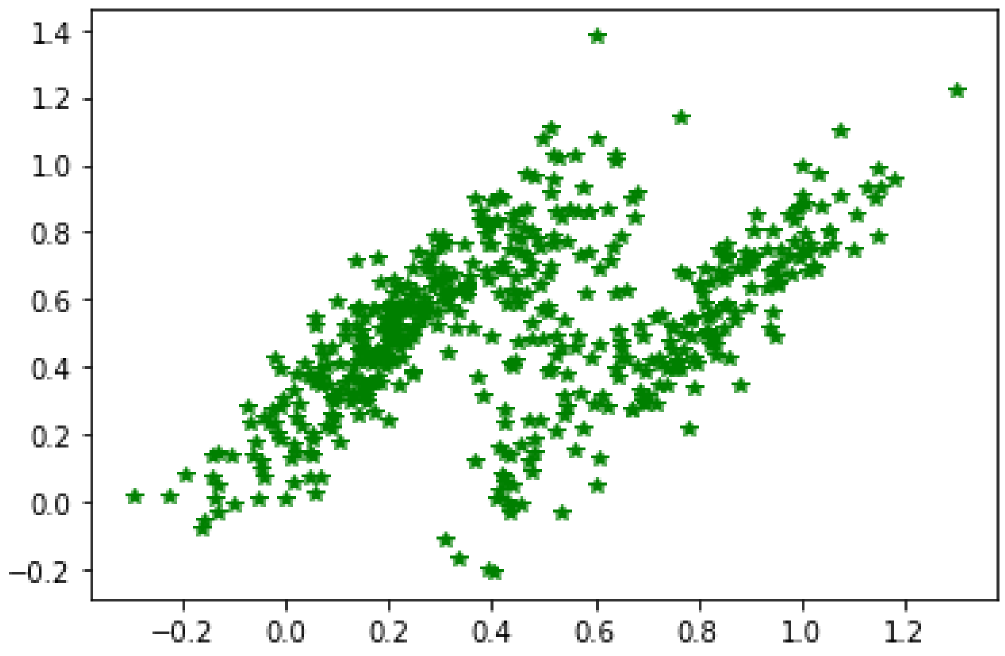
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5:
RuntimeWarning: covariance is not positive-semidefinite.
  """
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8:
RuntimeWarning: covariance is not positive-semidefinite.
```



OBSERVATIONS :-

In above fig, Samples from the joint distribution p(z)*p(x|z) in which the three states of z, corresponding to the three components of the mixture, are depicted in blue, green, and red.

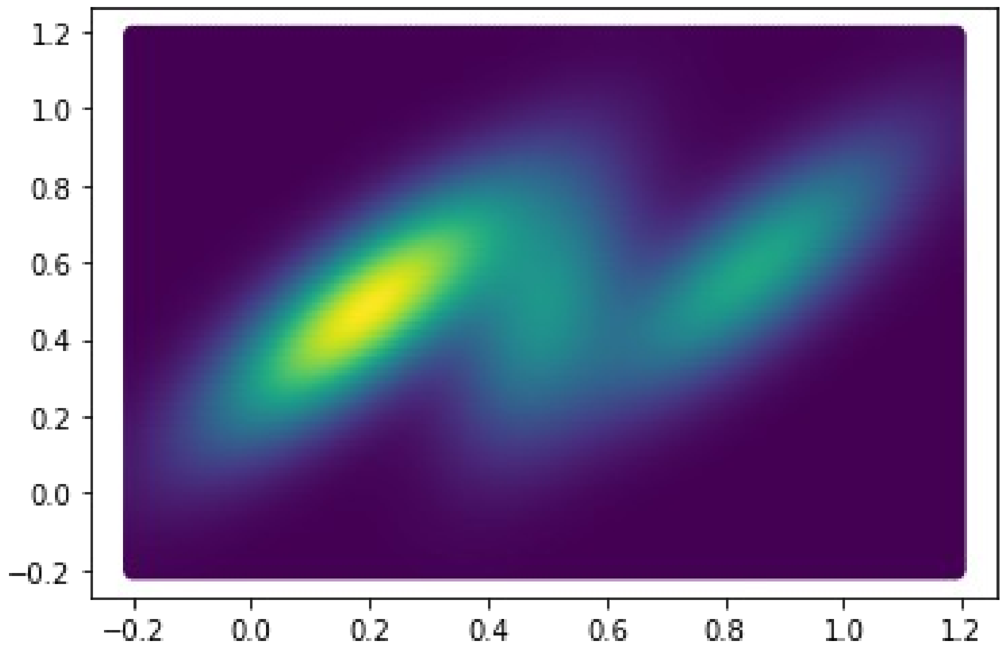
```
plt.plot(x1, y1, '*', color='green')
plt.plot(x2, y2, '*', color='green')
plt.plot(x3, y3, '*', color='green')
plt.show()
```



OBSERVATIONS :-

In above fig, corresponds the samples from the marginal distribution p(x), which is obtained by simply ignoring the values of z and just plotting the x values.

```
plt.scatter(x, y, c=pos)
plt.show()
```



OBSERVATIONS :-

In above fig, The same samples in which the colours represent the value of the
responsibilities γ(znk) associated with data point xn, obtained by plotting
the corresponding point using proportions of red, blue, and green ink given
by γ(znk) for k = 1, 2, 3, respectively

CONCLUSIONS :-

The darker region in plotting shows more points is taken from corresponding gaussian distribution and hence is having more hights in contour plot.