CS4044D: Machine Learning

Assignment 1

Name : Bhukya Vasanth Kumar
Roll : B180441CS
SEM - 7, CSE : A - Batch
Date : Nov 9, 2021

1) Discriminant function for the given normal density equation

Given normal density equation :

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^t\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\omega_i).$$

The Discriminant function given can be defined in the below form of code:

```
def discriminant_function(x, mean, cov, d, P):
   # Checking if the dimensions turn out to be scalars in the case
only 1 feature is being taken.
   if d == 1:
       output = -0.5*(x - mean) * (1/cov)
       output = output * (x - mean)
       output += -0.5*d*log(2*pi) - 0.5*log(cov)

   else:
       output = np.matmul(-0.5*(x - mean), np.linalg.inv(cov))
       output = np.matmul(output, (x - mean).T)
       output += -0.5*d*log(2*pi) - 0.5*log(np.linalg.det(cov))
   output += log(P)
   return output
```

The inputs given to the program for 3 classes:

```
data = [
# W1 CLASS
np.array([
    [-5.01, -8.12, -3.68],
    [-5.43, -3.48, -3.54],
    [1.08, -5.52, 1.66],
    [0.86, -3.78, -4.11],
    [-2.67, 0.63, 7.39],
    [4.94, 3.29, 2.08],
    [-2.51, 2.09, -2.59],
    [-2.25, -2.13, -6.94],
    [5.56, 2.86, -2.26],
    [1.03, -3.33, 4.33]
]),

# W2 CLASS
np.array([
    [-0.91, -0.18, -0.05],
    [1.30, -2.06, -3.53],
    [-7.75, -4.54, -0.95],
    [-5.47, 0.50, 3.92],
    [6.14, 5.72, -4.85],
    [3.60, 1.26, 4.36],
    [5.37, -4.63, -3.65],
    [7.18, 1.46, -6.66],
    [-7.39, 1.17, 6.30],
    [-7.50, -6.32, -0.31]

]),

# W3 CLASS
np.array([
    [5.35, 2.26, 8.13],
    [5.12, 3.22, -2.66],
    [-1.34, -5.31, -9.87],
    [4.48, 3.42, 5.19],
    [7.11, 2.39, 9.21],
    [7.17, 4.33, -0.98],
    [5.75, 3.97, 6.65],
            [0.77, 0.27, 2.41],
            [0.90, -0.43, -8.71],
            [3.52, -0.36, 6.43]
        ])
    ]
```

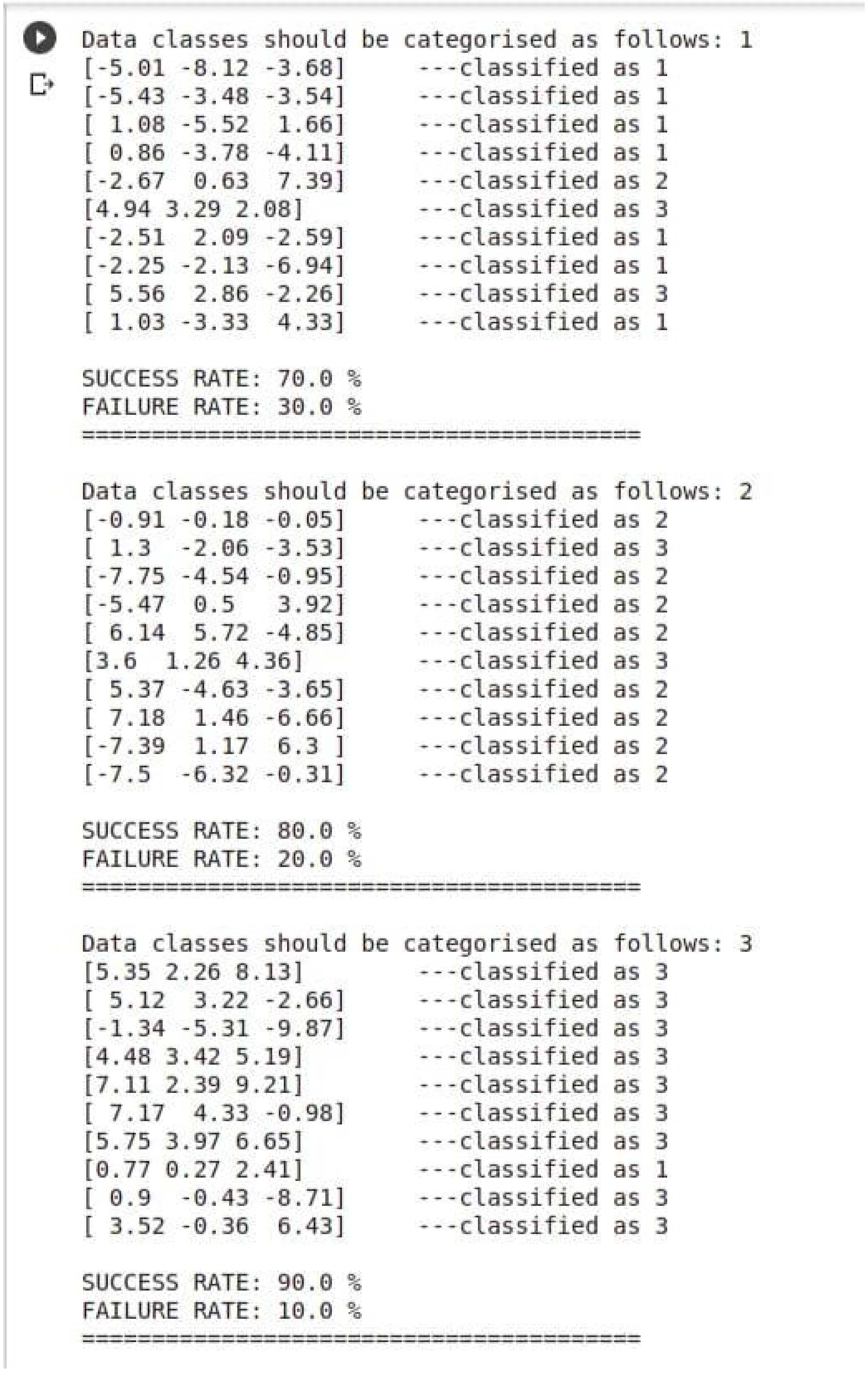
<u>Values for the Configuration</u>

```
    n = len(data)
    P = [1/n for i in range(n)]
    d = len(data[0][0])
```

To categorise the sample data, we must first run the function class wise through our dataset. On each sample, we look for the class with the highest discriminant function result. A count and total count are kept to calculate the success and failure rates.

```
      for j in range(n):
       print("\nData classes should be categorised as follows:", j+1)
       total_count, count = 0, 0

       for x in data[j]:
           g_values = [0 for g in range(n)]
           for i in range(n):
               g_values[i] = discriminant_function(x, means[i], cov[i], d,
P[i])

           result = g_values.index(max(g_values)) + 1
           print(x, "\t---classified as", result)
           total_count, count = total_count + 1, (count + 1 if j == result
- 1 else count)

       print("\nSUCCESS RATE:", (count/total_count)*100,"%")
       print("FAILURE RATE:", 100 - ((count/total_count))*100,"%")
       print("========================================")
```

OUTPUT (Assumption : Equal prior probability for all classes)



```
Data classes should be categorised as follows: 1
[-5.01 -8.12 -3.68]      ---classified as 1
[-5.43 -3.48 -3.54]      ---classified as 1
[ 1.08 -5.52  1.66]      ---classified as 1
[ 0.86 -3.78 -4.11]      ---classified as 1
[-2.67  0.63  7.39]      ---classified as 2
[4.94 3.29 2.08]         ---classified as 3
[-2.51  2.09 -2.59]      ---classified as 1
[-2.25 -2.13 -6.94]      ---classified as 1
[ 5.56  2.86 -2.26]      ---classified as 3
[ 1.03 -3.33  4.33]      ---classified as 1

SUCCESS RATE: 70.0 %
FAILURE RATE: 30.0 %
========================================

Data classes should be categorised as follows: 2
[-0.91 -0.18 -0.05]      ---classified as 2
[ 1.3  -2.06 -3.53]      ---classified as 3
[-7.75 -4.54 -0.95]      ---classified as 2
[-5.47  0.5   3.92]      ---classified as 2
[ 6.14  5.72 -4.85]      ---classified as 2
[3.6  1.26 4.36]         ---classified as 3
[ 5.37 -4.63 -3.65]      ---classified as 2
[ 7.18  1.46 -6.66]      ---classified as 2
[-7.39  1.17  6.3 ]      ---classified as 2
[-7.5  -6.32 -0.31]      ---classified as 2

SUCCESS RATE: 80.0 %
FAILURE RATE: 20.0 %
========================================

Data classes should be categorised as follows: 3
[5.35 2.26 8.13]         ---classified as 3
[ 5.12  3.22 -2.66]      ---classified as 3
[-1.34 -5.31 -9.87]      ---classified as 3
[4.48 3.42 5.19]         ---classified as 3
[7.11 2.39 9.21]         ---classified as 3
[ 7.17  4.33 -0.98]      ---classified as 3
[5.75 3.97 6.65]         ---classified as 3
[0.77 0.27 2.41]         ---classified as 1
[ 0.9  -0.43 -8.71]      ---classified as 3
[ 3.52 -0.36  6.43]      ---classified as 3

SUCCESS RATE: 90.0 %
FAILURE RATE: 10.0 %
========================================
```

2) Classifying 10 samples from the above question (1)

Solution: A

Designing a dichotomizer for those two categories using the feature x1 alone.
Assumption: The prior probabilities of the first two categories are equal and are equal to 1/2 and that of the third category is zero.

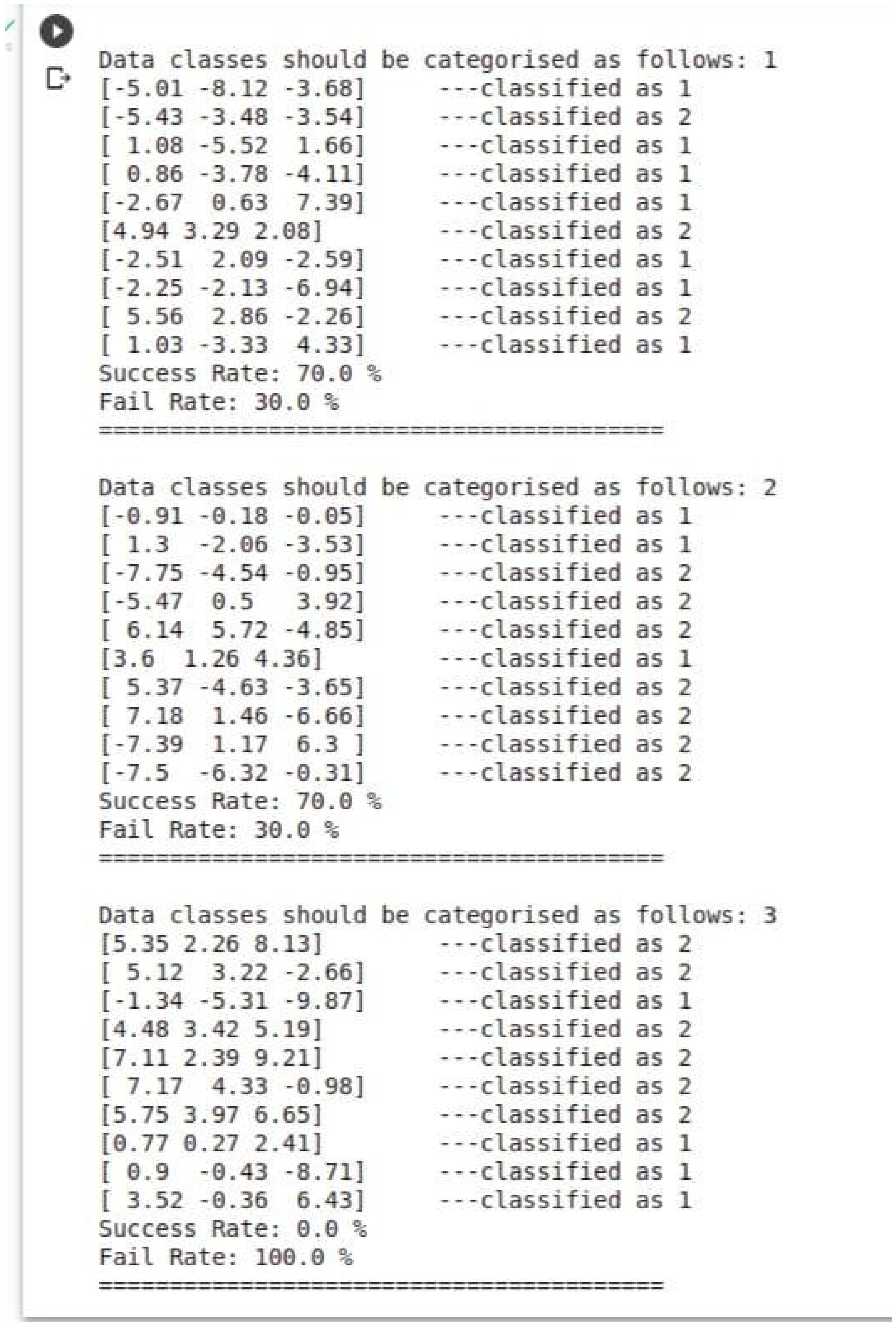
Configuration :

```
      n = len(data) - 1
      P = [0.5, 0.5, 0]
      d = 1
```

Iteration for the design :

```
   for j in range(n + 1):
       print("\nData classes should be categorised as follows:", j+1)
       total_count, count = 0, 0

       for x in data[j]:
           g_values = [0 for g in range(n)]

           for i in range(n):
               g_values[i] = discriminant_function(x[0], means[i][0],
cov[i][0][0], d, P[i])

           result = g_values.index(max(g_values)) + 1
           print(x, "\t---classified as", result)
           total_count, count = total_count + 1, (count + 1 if j == result
- 1 else count)

       print("Success Rate:", (count/total_count)*100,"%")
       print("Fail Rate:", 100 - ((count/total_count))*100,"%")
       print("========================================")
```

<u>Solution: B:</u> Percentage of points misclassified (**OUTPUT**)



```
Data classes should be categorised as follows: 1
[-5.01 -8.12 -3.68]     ---classified as 1
[-5.43 -3.48 -3.54]     ---classified as 2
[ 1.08 -5.52  1.66]     ---classified as 1
[ 0.86 -3.78 -4.11]     ---classified as 1
[-2.67  0.63  7.39]     ---classified as 1
[4.94 3.29 2.08]        ---classified as 2
[-2.51  2.09 -2.59]     ---classified as 1
[-2.25 -2.13 -6.94]     ---classified as 1
[ 5.56  2.86 -2.26]     ---classified as 2
[ 1.03 -3.33  4.33]     ---classified as 1
Success Rate: 70.0 %
Fail Rate: 30.0 %
========================================

Data classes should be categorised as follows: 2
[-0.91 -0.18 -0.05]     ---classified as 1
[ 1.3  -2.06 -3.53]     ---classified as 1
[-7.75 -4.54 -0.95]     ---classified as 2
[-5.47  0.5   3.92]     ---classified as 2
[ 6.14  5.72 -4.85]     ---classified as 2
[3.6  1.26 4.36]        ---classified as 1
[ 5.37 -4.63 -3.65]     ---classified as 2
[ 7.18  1.46 -6.66]     ---classified as 2
[-7.39  1.17  6.3 ]     ---classified as 2
[-7.5  -6.32 -0.31]     ---classified as 2
Success Rate: 70.0 %
Fail Rate: 30.0 %
========================================

Data classes should be categorised as follows: 3
[5.35 2.26 8.13]        ---classified as 2
[ 5.12  3.22 -2.66]     ---classified as 2
[-1.34 -5.31 -9.87]     ---classified as 1
[4.48 3.42 5.19]        ---classified as 2
[7.11 2.39 9.21]        ---classified as 2
[ 7.17  4.33 -0.98]     ---classified as 2
[5.75 3.97 6.65]        ---classified as 2
[0.77 0.27 2.41]        ---classified as 1
[ 0.9  -0.43 -8.71]     ---classified as 1
[ 3.52 -0.36  6.43]     ---classified as 1
Success Rate: 0.0 %
Fail Rate: 100.0 %
========================================
```

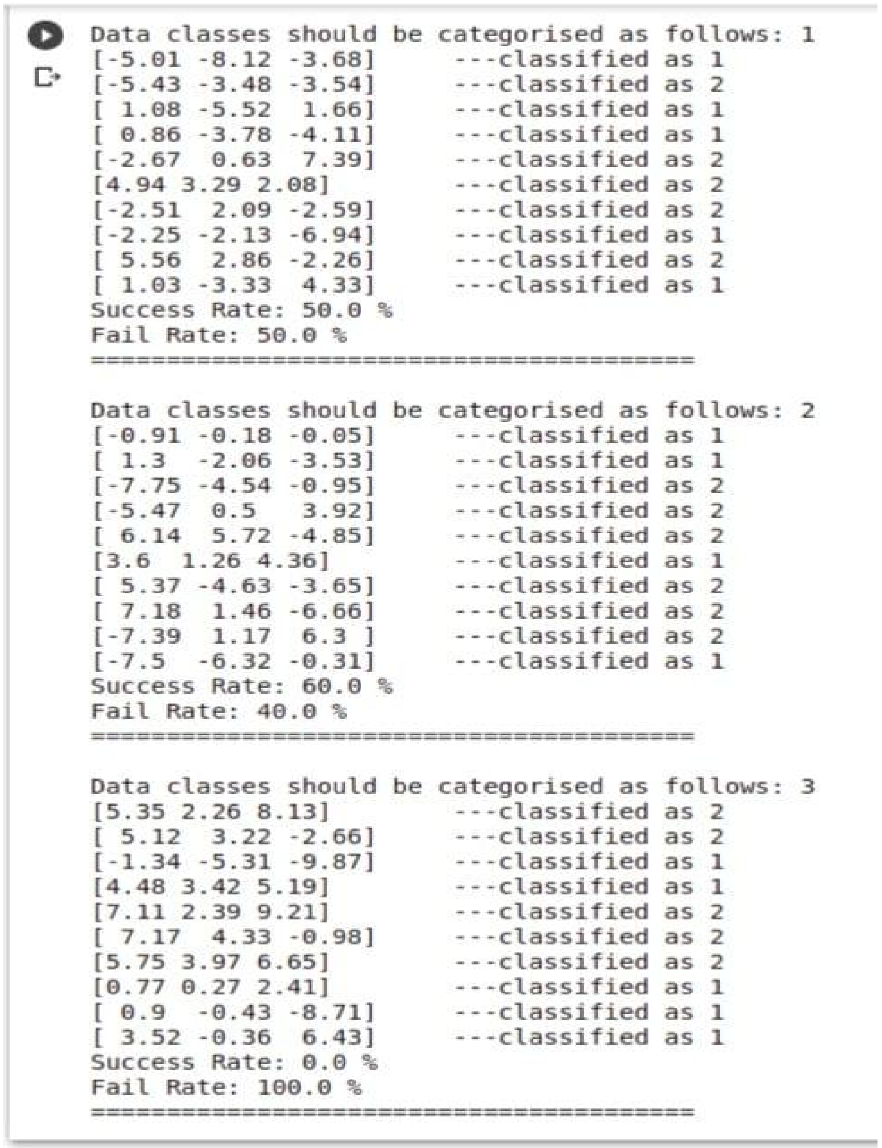
Solution: C

The only change is to use the two features x1 and x2. Thus, the value of d is modified to 2. Matrix parameters are modified.

Configuration :

```
n = len(data) - 1
P = [0.5, 0.5, 0]
d = 2
```

OUTPUT



```
Data classes should be categorised as follows: 1
[-5.01 -8.12 -3.68]      ---classified as 1
[-5.43 -3.48 -3.54]      ---classified as 2
[ 1.08 -5.52  1.66]      ---classified as 1
[ 0.86 -3.78 -4.11]      ---classified as 1
[-2.67  0.63  7.39]      ---classified as 2
[4.94 3.29 2.08]         ---classified as 2
[-2.51  2.09 -2.59]      ---classified as 2
[-2.25 -2.13 -6.94]      ---classified as 1
[ 5.56  2.86 -2.26]      ---classified as 2
[ 1.03 -3.33  4.33]      ---classified as 1
Success Rate: 50.0 %
Fail Rate: 50.0 %
========================================

Data classes should be categorised as follows: 2
[-0.91 -0.18 -0.05]      ---classified as 1
[ 1.3  -2.06 -3.53]      ---classified as 1
[-7.75 -4.54 -0.95]      ---classified as 2
[-5.47  0.5   3.92]      ---classified as 2
[ 6.14  5.72 -4.85]      ---classified as 2
[3.6  1.26 4.36]         ---classified as 1
[ 5.37 -4.63 -3.65]      ---classified as 2
[ 7.18  1.46 -6.66]      ---classified as 2
[-7.39  1.17  6.3 ]      ---classified as 2
[-7.5  -6.32 -0.31]      ---classified as 1
Success Rate: 60.0 %
Fail Rate: 40.0 %
========================================

Data classes should be categorised as follows: 3
[5.35 2.26 8.13]         ---classified as 2
[ 5.12  3.22 -2.66]      ---classified as 2
[-1.34 -5.31 -9.87]      ---classified as 1
[4.48 3.42 5.19]         ---classified as 1
[7.11 2.39 9.21]         ---classified as 2
[ 7.17  4.33 -0.98]      ---classified as 2
[5.75 3.97 6.65]         ---classified as 2
[0.77 0.27 2.41]         ---classified as 1
[ 0.9  -0.43 -8.71]      ---classified as 1
[ 3.52 -0.36  6.43]      ---classified as 1
Success Rate: 0.0 %
Fail Rate: 100.0 %
========================================
```

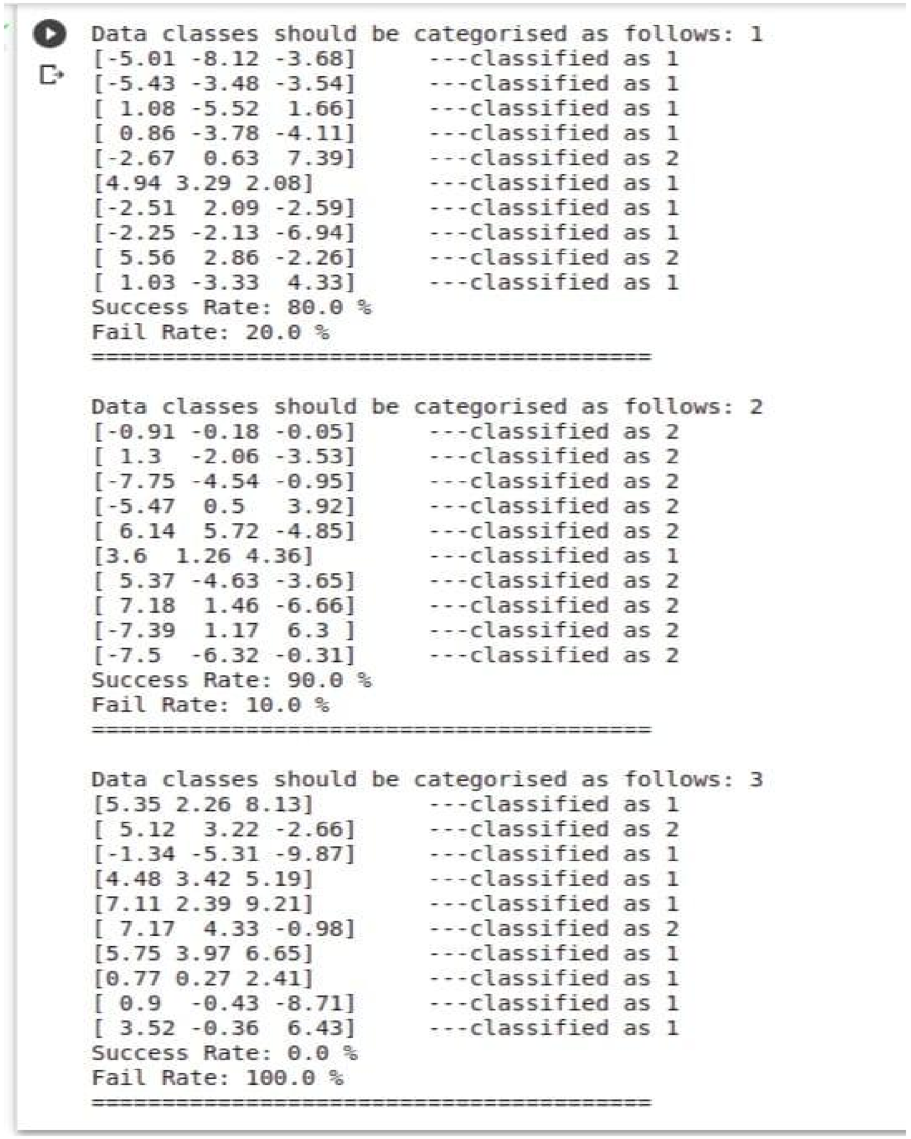
Solution: D

The only change is to use the two features x1 and x2. Thus, the value of d is modified to 3. Matrix parameters are modified.

Configuration :

```
n = len(data) - 1
P = [0.5, 0.5, 0]
d = 3
```

OUTPUT



```
Data classes should be categorised as follows: 1
[-5.01 -8.12 -3.68]      ---classified as 1
[-5.43 -3.48 -3.54]      ---classified as 1
[ 1.08 -5.52  1.66]      ---classified as 1
[ 0.86 -3.78 -4.11]      ---classified as 1
[-2.67  0.63  7.39]      ---classified as 2
[4.94 3.29 2.08]         ---classified as 1
[-2.51  2.09 -2.59]      ---classified as 1
[-2.25 -2.13 -6.94]      ---classified as 1
[ 5.56  2.86 -2.26]      ---classified as 2
[ 1.03 -3.33  4.33]      ---classified as 1
Success Rate: 80.0 %
Fail Rate: 20.0 %
========================================

Data classes should be categorised as follows: 2
[-0.91 -0.18 -0.05]      ---classified as 2
[ 1.3  -2.06 -3.53]      ---classified as 2
[-7.75 -4.54 -0.95]      ---classified as 2
[-5.47  0.5   3.92]      ---classified as 2
[ 6.14  5.72 -4.85]      ---classified as 2
[3.6  1.26 4.36]         ---classified as 1
[ 5.37 -4.63 -3.65]      ---classified as 2
[ 7.18  1.46 -6.66]      ---classified as 2
[-7.39  1.17  6.3 ]      ---classified as 2
[-7.5  -6.32 -0.31]      ---classified as 2
Success Rate: 90.0 %
Fail Rate: 10.0 %
========================================

Data classes should be categorised as follows: 3
[5.35 2.26 8.13]         ---classified as 1
[ 5.12  3.22 -2.66]      ---classified as 2
[-1.34 -5.31 -9.87]      ---classified as 1
[4.48 3.42 5.19]         ---classified as 1
[7.11 2.39 9.21]         ---classified as 1
[ 7.17  4.33 -0.98]      ---classified as 2
[5.75 3.97 6.65]         ---classified as 1
[0.77 0.27 2.41]         ---classified as 1
[ 0.9  -0.43 -8.71]      ---classified as 1
[ 3.52 -0.36  6.43]      ---classified as 1
Success Rate: 0.0 %
Fail Rate: 100.0 %
========================================
```

Solution: E

After comparing the results obtained, it can be concluded that using one or three features give more accurate results than using the first and second features. This could be due to the fact that the covariance associated with the third feature is substantially higher than those connected with the second feature.

Solution: F

Have to classify the points (1,2,1)t,, (5,3,2)t , (0,0,0)t , (1,0,0)t using each feature vector mentioned above.

There can be 3 divisions here,

1. Using feature vector x1
2. Using feature vectors x1 and x2
3. Using all the given feature vectors

Configuration

```
n = len(data) - 1
P = [0.5, 0.5, 0]
g_values = [0 for i in range(n)]
```

Division 1

```
d = 1
print("Division 1: Using feature vector x1")
for i in range(n):
    g_values[i] = discriminant_function(x[0], means[i][0],
cov[i][0][0], d, P[i])
result = g_values.index(max(g_values)) + 1
print(x, "\t---classified as", result)
```

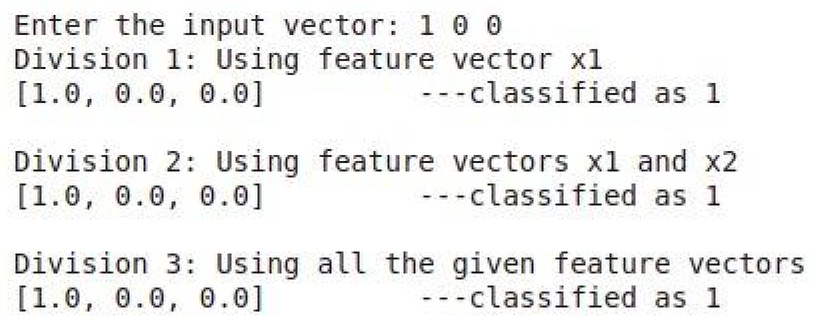
Division 2

```
   d = 2
   print("\nDivision 2: Using feature vectors x1 and x2")
   for i in range(n):
       g_values[i] = discriminant_function(x[0:2], means[i][0:2],
cov[i][0:2, 0:2], d, P[i])
   result = g_values.index(max(g_values)) + 1
   print(x, "\t---classified as", result)
```

Division 3

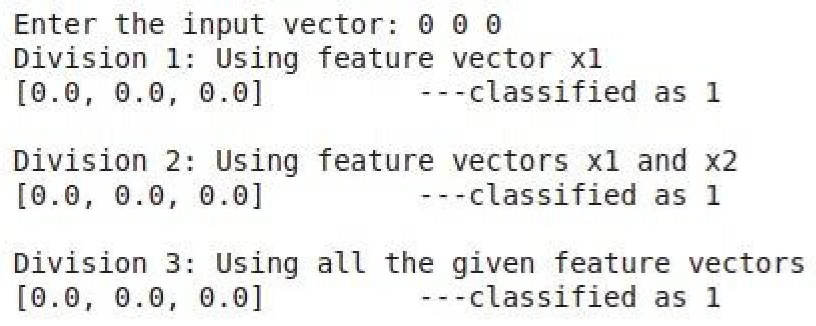
```
   d = 3
   print("\nDivision 3: Using all the given feature vectors")
   for i in range(n):
       g_values[i] = discriminant_function(x, means[i], cov[i], d, P[i])
   result = g_values.index(max(g_values)) + 1
   print(x, "\t---classified as", result)
```

OUTPUT FOR THE GIVEN POINTS

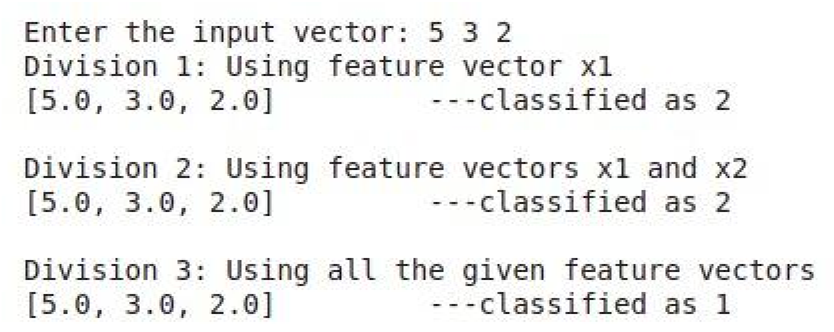


```
Enter the input vector: 1 0 0
Division 1: Using feature vector x1
[1.0, 0.0, 0.0]         ---classified as 1

Division 2: Using feature vectors x1 and x2
[1.0, 0.0, 0.0]         ---classified as 1

Division 3: Using all the given feature vectors
[1.0, 0.0, 0.0]         ---classified as 1
```

==

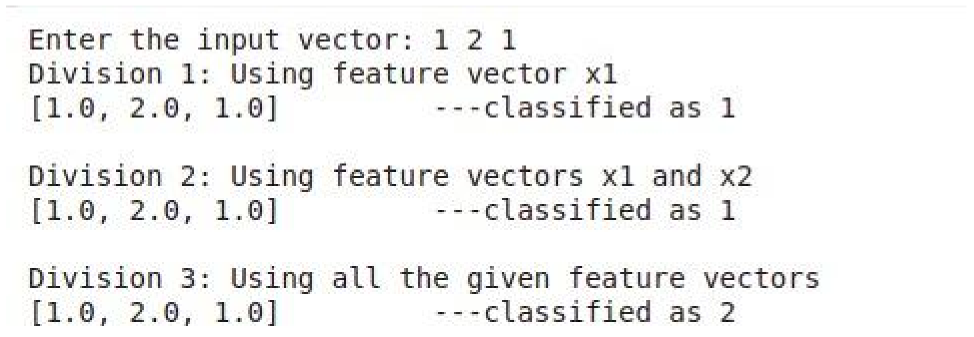


```
Enter the input vector: 0 0 0
Division 1: Using feature vector x1
[0.0, 0.0, 0.0]         ---classified as 1

Division 2: Using feature vectors x1 and x2
[0.0, 0.0, 0.0]         ---classified as 1

Division 3: Using all the given feature vectors
[0.0, 0.0, 0.0]         ---classified as 1
```



```
Enter the input vector: 5 3 2
Division 1: Using feature vector x1
[5.0, 3.0, 2.0]         ---classified as 2

Division 2: Using feature vectors x1 and x2
[5.0, 3.0, 2.0]         ---classified as 2

Division 3: Using all the given feature vectors
[5.0, 3.0, 2.0]         ---classified as 1
```

==



```
Enter the input vector: 1 2 1
Division 1: Using feature vector x1
[1.0, 2.0, 1.0]         ---classified as 1

Division 2: Using feature vectors x1 and x2
[1.0, 2.0, 1.0]         ---classified as 1

Division 3: Using all the given feature vectors
[1.0, 2.0, 1.0]         ---classified as 2
```