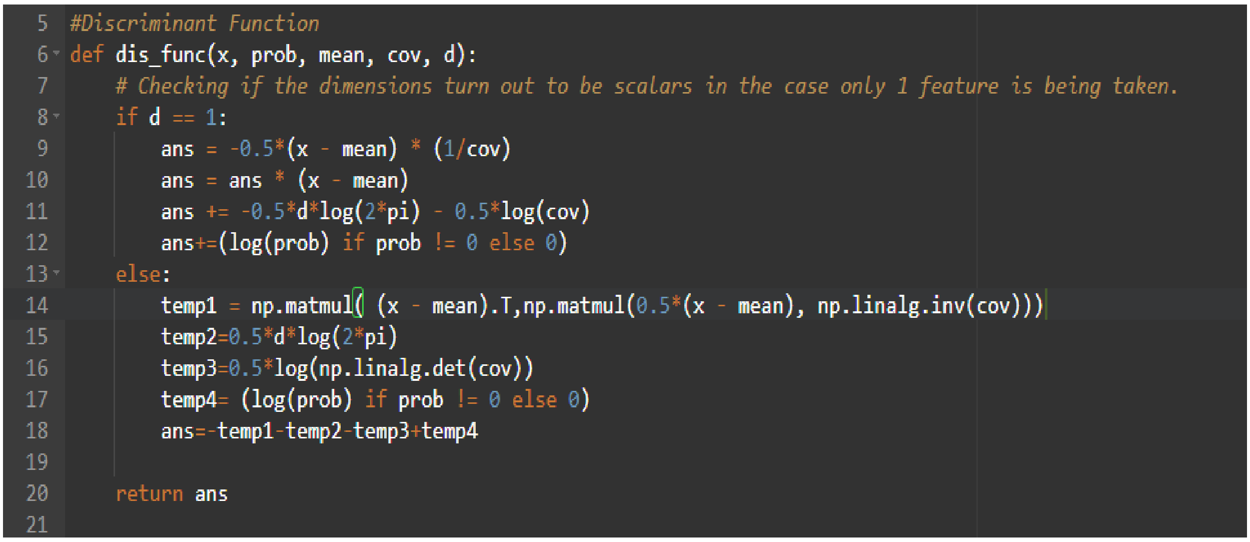
Problem 2 –

Consider the problem of classifying 10 samples from the above table of data. Assume the that the underlying distributions are normal.

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

2.b. Determine the percentage of points misclassified.

Solution (2a, 2b) –



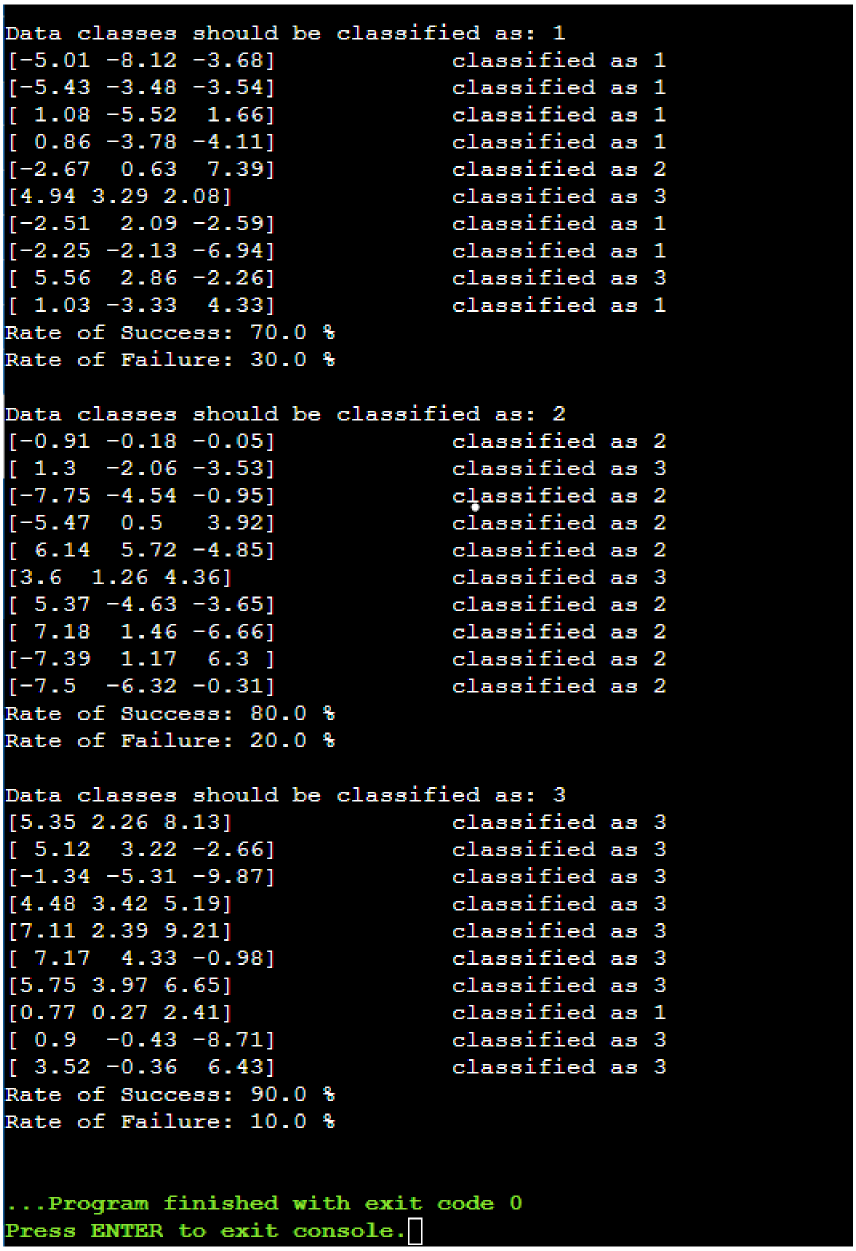
```
#Discriminant Function
def dis_func(x, prob, mean, cov, d):
    # Checking if the dimensions turn out to be scalars in the case only 1 feature is being taken.
    if d == 1:
        ans = -0.5*(x - mean) * (1/cov)
        ans = ans * (x - mean)
        ans += -0.5*d*log(2*pi) - 0.5*log(cov)
        ans+=(log(prob) if prob != 0 else 0)
    else:
        temp1 = np.matmul( (x - mean).T,np.matmul(0.5*(x - mean), np.linalg.inv(cov)))
        temp2=0.5*d*log(2*pi)
        temp3=0.5*log(np.linalg.det(cov))
        temp4= (log(prob) if prob != 0 else 0)
        ans=-temp1-temp2-temp3+temp4

    return ans
```

Here discriminant function has been modified to include the case where dimension is 1.

Some changes in configuration of input data

- d=1 indicates here that only one feature x1 is used.
- x[0] indicates that only x_1 will be used.
- means[i][0] indiciates that we need the mean only for x_1.
- cov[i][0][0] indicates the variance of feature x_1.

Output:-



```
Data classes should be classified as: 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
Rate of Success: 70.0 %
Rate of Failure: 30.0 %

Data classes should be classified as: 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
Rate of Success: 80.0 %
Rate of Failure: 20.0 %

Data classes should be classified as: 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
Rate of Success: 90.0 %
Rate of Failure: 10.0 %


...Program finished with exit code 0
Press ENTER to exit console.
```

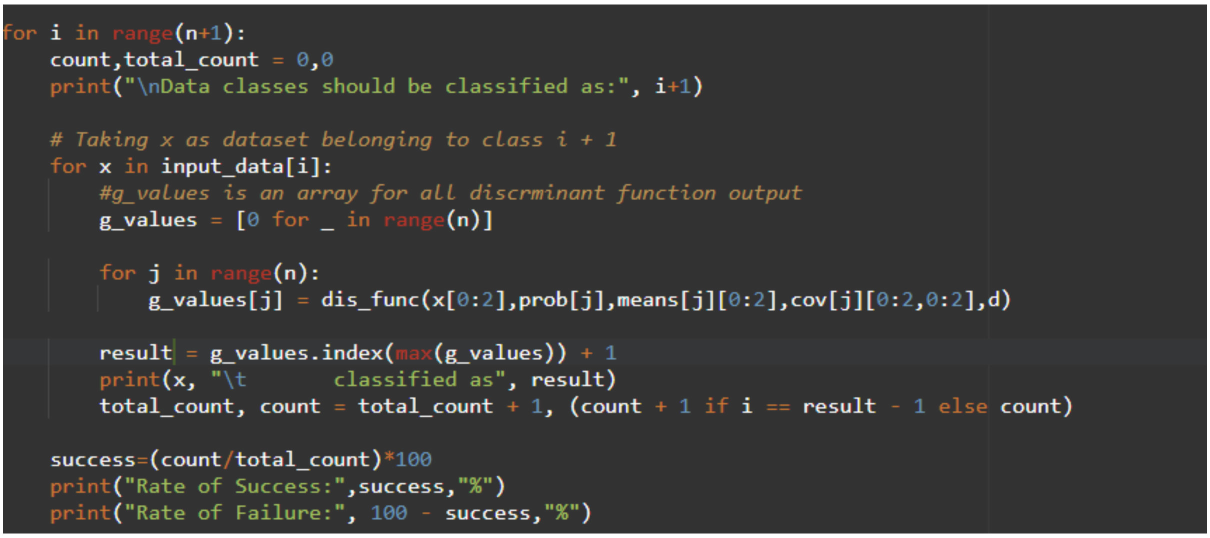
2.c. Repeat the above two steps, but now use the two features x1 and x2.

Solution :-

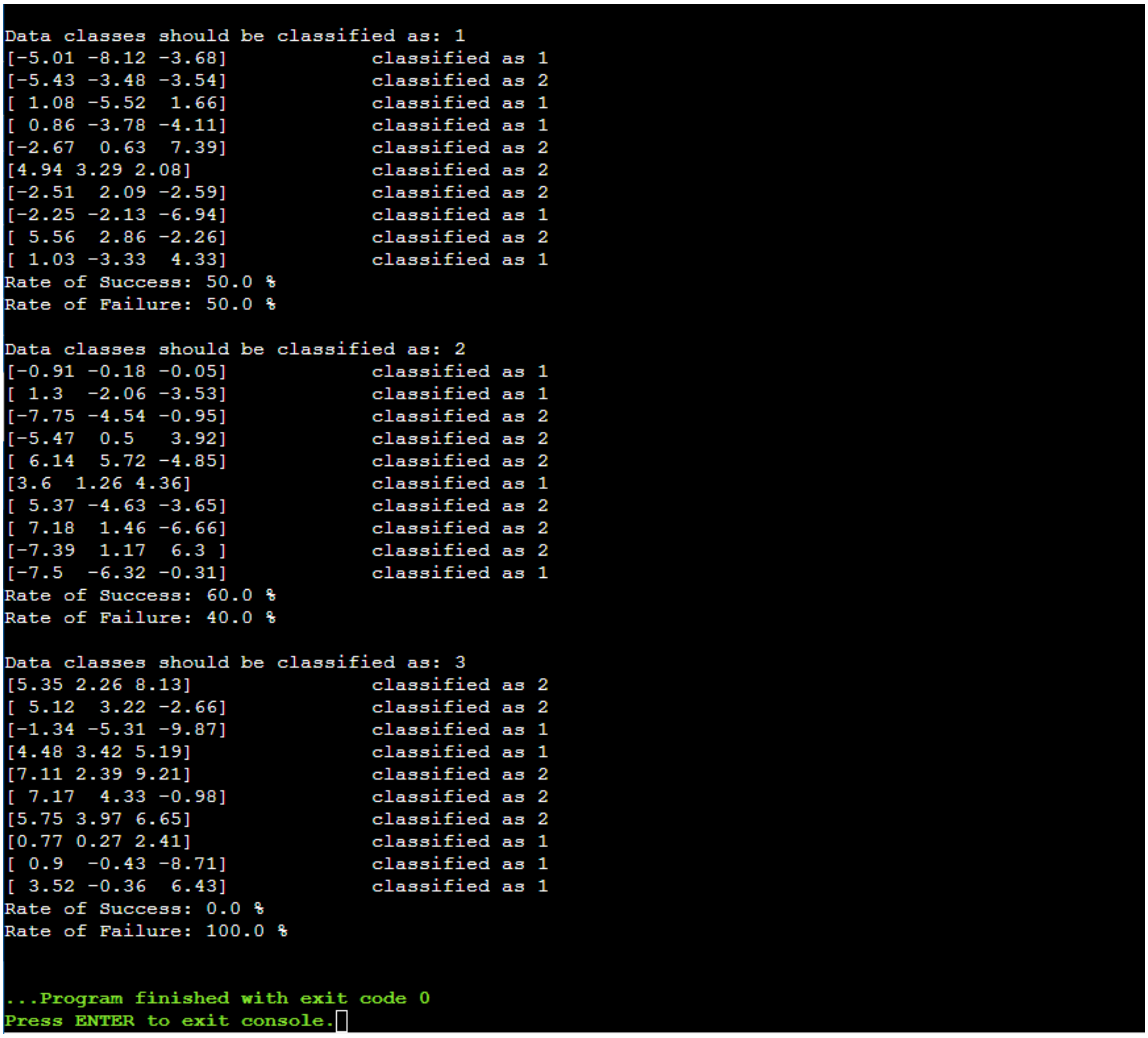
Input is such that it considers 2 features

d = 2 for considering 2 features

Output -



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

    # Taking x as dataset belonging to class i + 1
    for x in input_data[i]:
        #g_values is an array for all discrminant function output
        g_values = [0 for _ in range(n)]

        for j in range(n):
            g_values[j] = dis_func(x[0:2],prob[j],means[j][0:2],cov[j][0:2,0:2],d)

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

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



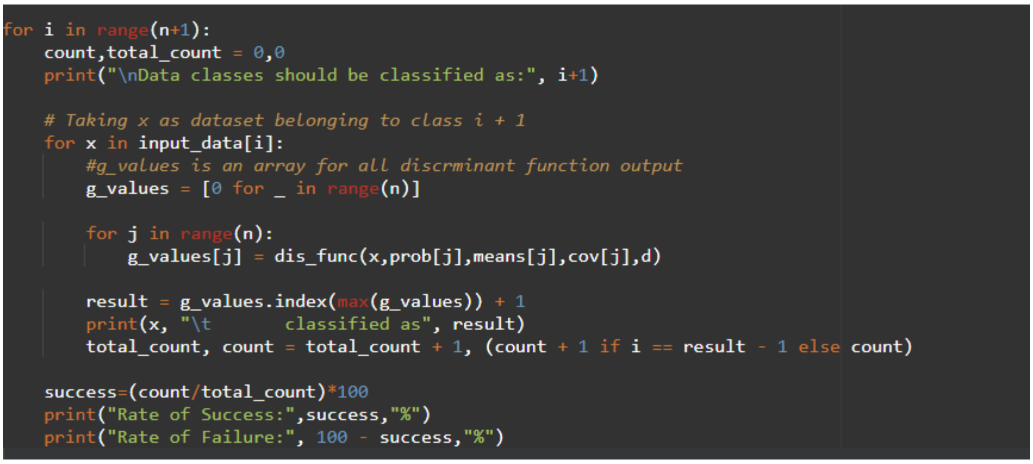
```
Data classes should be classified as: 1
[-5.01 -8.12 -3.68]            classified as 1
[-5.43 -3.48 -3.54]            classified as 2
[ 1.08 -5.52  1.66]            classified as 1
[ 0.86 -3.78 -4.11]            classified as 1
[-2.67  0.63  7.39]            classified as 2
[4.94 3.29 2.08]               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
Rate of Success: 50.0 %
Rate of Failure: 50.0 %

Data classes should be classified as: 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
Rate of Success: 60.0 %
Rate of Failure: 40.0 %

Data classes should be classified as: 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
Rate of Success: 0.0 %
Rate of Failure: 100.0 %


...Program finished with exit code 0
Press ENTER to exit console.
```

2.d. Repeat again, with all the three features taken.

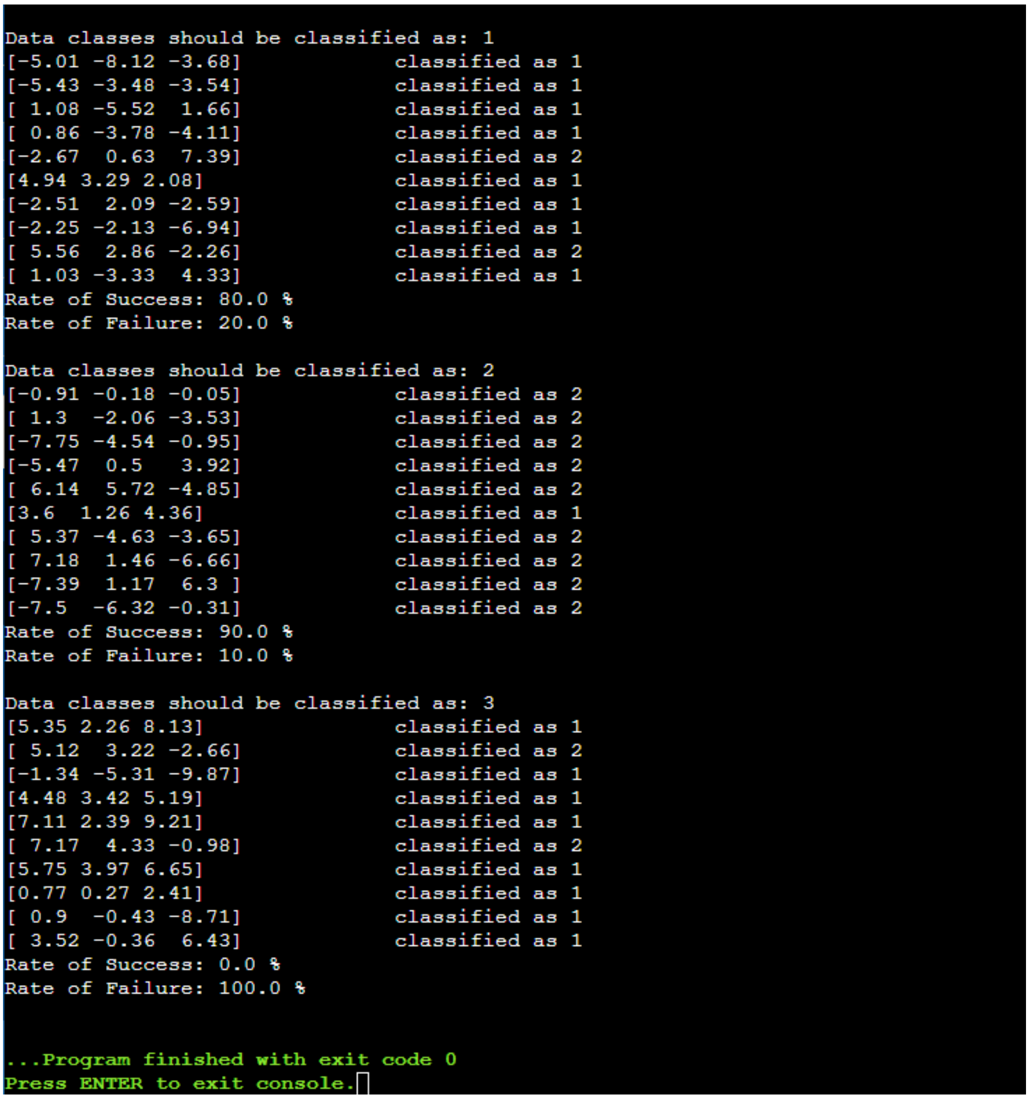
Solution :-

d = 3 for considering all features

Output :-



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

    # Taking x as dataset belonging to class i + 1
    for x in input_data[i]:
        #g_values is an array for all discrminant function output
        g_values = [0 for _ in range(n)]

        for j in range(n):
            g_values[j] = dis_func(x,prob[j],means[j],cov[j],d)

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

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



```
Data classes should be classified as: 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
Rate of Success: 80.0 %
Rate of Failure: 20.0 %

Data classes should be classified as: 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
Rate of Success: 90.0 %
Rate of Failure: 10.0 %

Data classes should be classified as: 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
Rate of Success: 0.0 %
Rate of Failure: 100.0 %


...Program finished with exit code 0
Press ENTER to exit console.
```

2.e. Compare your results and conclude.

Solution :-

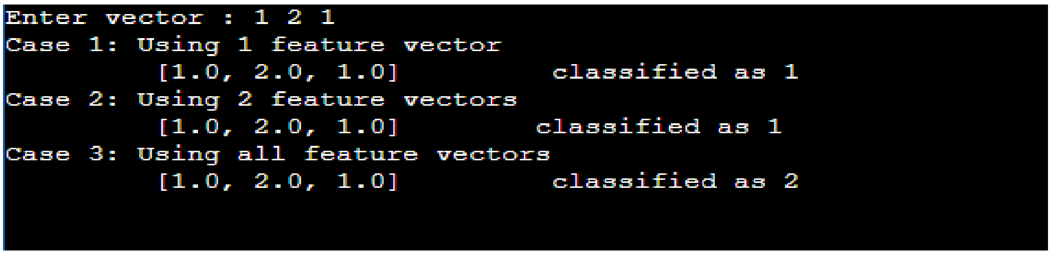
On comparing the three outputs, using one or three features give more accurate results than using the first and second features.

2.f. Classify the points (1,2,1), (5,3,2), (0,0,0), (1,0,0) using each feature vector mentioned above and compare the results.

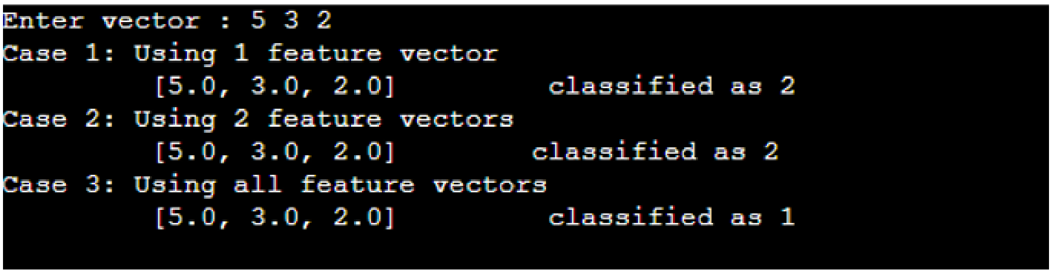
Solution :-

We have taken all vectors as inputs and applied the discriminant function with dimension 1,2 and 3 as three cases.

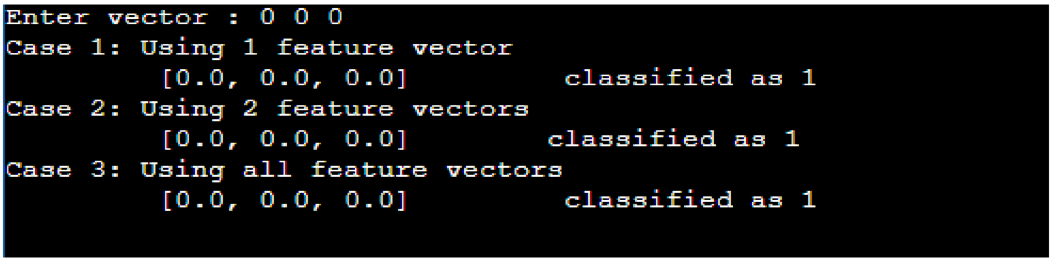
Output:-



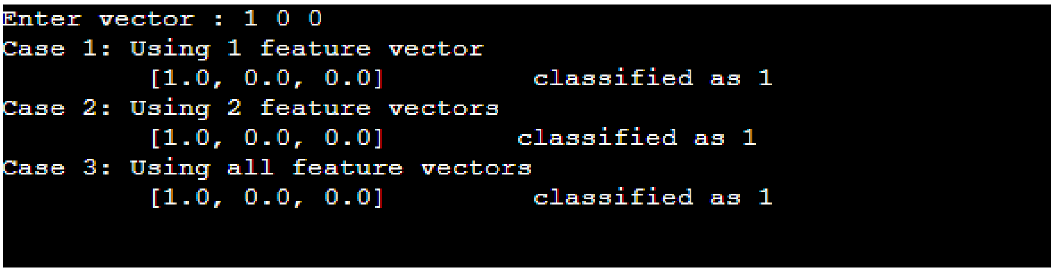
```
Enter vector : 1 2 1
Case 1: Using 1 feature vector
         [1.0, 2.0, 1.0]         classified as 1
Case 2: Using 2 feature vectors
         [1.0, 2.0, 1.0]        classified as 1
Case 3: Using all feature vectors
         [1.0, 2.0, 1.0]         classified as 2
```



```
Enter vector : 5 3 2
Case 1: Using 1 feature vector
         [5.0, 3.0, 2.0]         classified as 2
Case 2: Using 2 feature vectors
         [5.0, 3.0, 2.0]        classified as 2
Case 3: Using all feature vectors
         [5.0, 3.0, 2.0]         classified as 1
```



```
Enter vector : 0 0 0
Case 1: Using 1 feature vector
         [0.0, 0.0, 0.0]         classified as 1
Case 2: Using 2 feature vectors
         [0.0, 0.0, 0.0]        classified as 1
Case 3: Using all feature vectors
         [0.0, 0.0, 0.0]         classified as 1
```



```
Enter vector : 1 0 0
Case 1: Using 1 feature vector
         [1.0, 0.0, 0.0]         classified as 1
Case 2: Using 2 feature vectors
         [1.0, 0.0, 0.0]        classified as 1
Case 3: Using all feature vectors
         [1.0, 0.0, 0.0]         classified as 1
```