ENGR-421 HW-1 REPORT

Name-Surname: Barış KAPLAN

Initially, I have imported the necessary libraries. The necessary libraries are as follows:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
```

Then, I have created the needed data by using array method of numpy. I have used the array method of numpy for the covariance values of the class, mean values of the class, and the size values of the class.

After that, I have generated the random points(by utilizing random.multivariate_normal method of numpy) and concatenated these random points(by utilizing vstack method of numpy)

I have generated the labels of the classes with the following code:

```
conc = np.concatenate((np.repeat(1, sizeValsOfTheClass[0]), np.repeat(2, sizeValsOfTheClass[1]), np.repeat(3, sizeValsOfTheClass[2])))
```

Subsequently, I have plotted the random data. While plotting, I have used plt.figure, plt.plot, plt.xlabel, plt.ylabel, and plt.show methods. The generated plot of the random data is as follows:

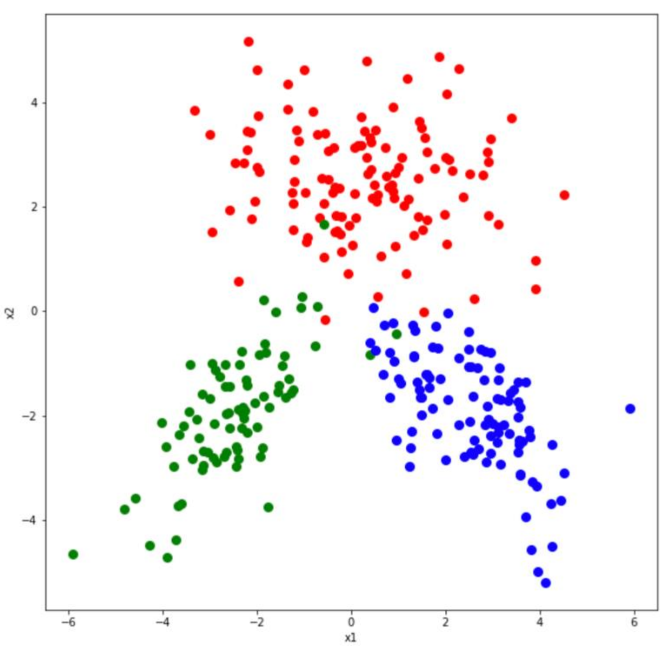

Figure 1: Generated plot

After that, prior probabilities as follows:

For Means

In the estimation of the means, I have initially defined a findMaximum function which finds the maximum value in an array. I have called this function with the parameter "conc". I have written the following code to estimate the mean parameters:

max_val= findMaximum(conc)
sample_mean_value_estimation_interval = [np.mean(stackOfRandomPoints[conc == (num + 1)], axis=0) for num in range(max_val)]

While estimating the mean parameters, I have used the following formula:

$$\hat{\mu}_c = \frac{\sum_{i=1}^{N}\left[1(y_i = c)\cdot x_i\right]}{N_c}$$

Figure 2: The formula for the mean parameter estimation

Note: In the above formula, Nc represents the number of data points in the class c.

After that, by utilizing np.array method, I have concatenated the 1st, 2nd, and 3rd element of the sample_mean_value_estimation_interval array.

Mean parameters estimation results:

```
Sample Mean Value Estimation Intervals:

1st class:

The sample mean value estimation interval for class1 is:
 [0.26563078 2.52716835]

2nd class:

The sample mean value estimation interval for class2 is:
 [-2.47809683 -1.95300192]

3rd class:

The sample mean value estimation interval for class3 is:
 [ 2.56009664 -1.90556717]

Sample Mean Est:

[[ 0.26563078  2.52716835]
 [-2.47809683 -1.95300192]
 [ 2.56009664 -1.90556717]]
```

Figure 3: The results of the mean estimations

s:

exCovarianceMatrices = [np.matmul(np.transpose(stackOfRandomPoints[conc == (num2+1)] - sample_mean_value_estimation_interval[num2]),stackOfRandomPoints[conc == (num2+1)] - sample_mean_value_estimation_interval[num2])/sizeValsOfTheClass[num2] for num2 in range(max_val)]

While estimating the covariance matrix parameters, I have used the following formula:

$$\hat{\Sigma}_{D\times D} = \frac{\sum_{i=1}^{N}(x_i-\hat{\mu})(x_i-\hat{\mu})^T}{N}$$

Sample Covariance matrix

Figure 4: The formula used for the sample covariance matrix estimation

```
Sample Covariance Matrices:

Covariance Matrix for the first class:
 [[ 2.66148669 -0.23424653]
 [-0.23424653  1.16477455]]

Covariance Matrix for the second class:
 [[1.19581994 0.92481706]
 [0.92481706 1.39635535]]

Covariance Matrix for the third class:
 [[ 1.23731467 -0.70749062]
 [-0.70749062  1.13882277]]
```

Figure 5: The results of the covariance matrix estimations

<u>For Class Prior Probabilities</u>

I have written the below code to estimate the class prior probability parameters:

class_pr_probs = [np.mean(conc == (my_num + 1)) for my_num in range(max_val)]
print(class_pr_probs)

While estimating the class prior probability parameters, I have used the following formula:

$$\hat{P}(y=c) = \frac{\sum_{i=1}^{N} 1(y_i=c)}{N}$$

Figure 6: The formula used for the class prior probability parameter estimation

```
[0.4, 0.26666666666666666, 0.3333333333333333]
```

Figure 7: The result of the prior probability estimations

Next, I have calculated the score values of the classes by utilizing the following formula:

$$-\frac{D}{2}\log(2\pi) - \frac{1}{2}\log(|\hat{\Sigma}_c|) - \frac{1}{2}(x-\hat{\mu}_c)^T \cdot \hat{\Sigma}_c^{-1} \cdot (x-\hat{\mu}_c) + \log[\hat{P}(y=c)]$$

ix with the following code:

['y_pred'], colnames = ['y_truth'])

Figure 8: The formula which is used for calculating the score values of the classes

print()
print(cnf_mat)

```
The Confusion Matrix:


y_truth    1   2    3
y_pred
1        116   1    0
2          1  77    0
3          3   2  100
```

Figure 9: Generated Confusion matrix

Next, I have drawn the decision boundaries. While drawing the decision boundaries, I have used plt.figure, plt.plot , and plt.contour functions. We can conclude from the confusion matrix that there exists 7 misclassified data points in total. There exist 4 (1+3) misclassified data points for the first class (having red color) , 3 (1+2) misclassified data points for the second class (having green color) , and 0 misclassified data points (having blue color) for the third class. These misclassified data points can be observed from Figure 10.

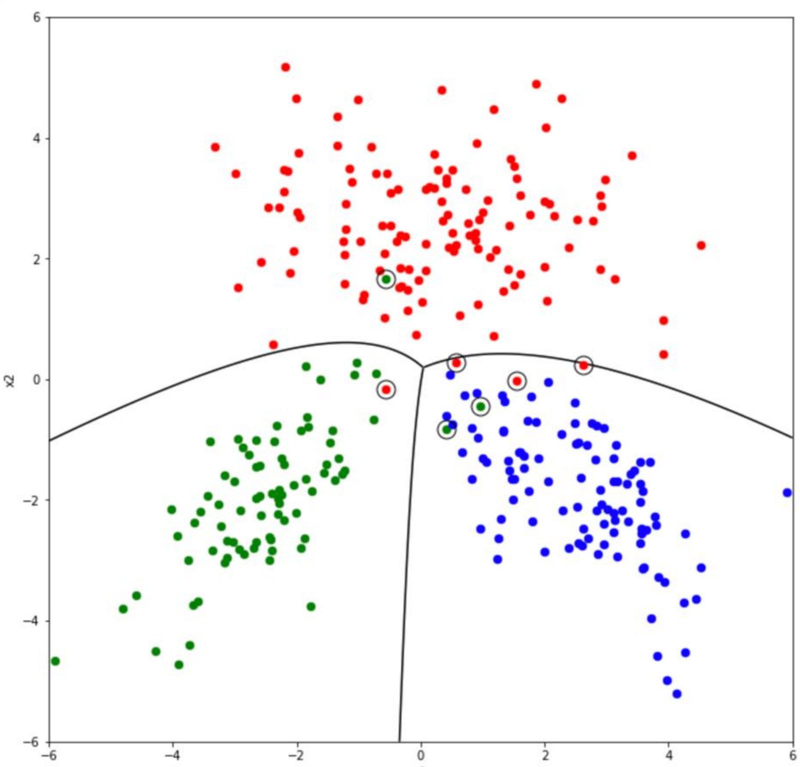

Figure 10: The plot including the decision boundaries and the misclassified data points