**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])))**

**Chart, scatter chart

Description automatically generatedSubsequently, 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, I have estimated the means, covariance matrices, and class 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:**

**Text, whiteboard

Description automatically generated**

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:**

**Graphical user interface, text, application

Description automatically generated**

Figure 3: The results of the mean estimations

**For Covariance Matrices**

**I have written the following to estimate the covariance matrices:**

**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:**

**Text

Description automatically generatedText

Description automatically generated**

Figure 5: The results of the covariance matrix estimations

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

**For Class Prior Probabilities**

**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:**

**Text

Description automatically generated**

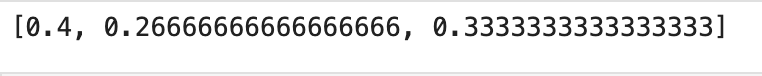
****

Figure 7: The result of the prior probability estimations

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

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

**Text

Description automatically generated**

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

**Subsequently, I have generated the confusion matrix with the following code:**

**cnf\_mat = pd.crosstab(concPred, conc, rownames = ['y\_pred'], colnames = ['y\_truth'])**

**print("The Confusion Matrix: \n")**

**Table

Description automatically generated with medium confidenceprint()**

**print(cnf\_mat)**

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.**

**Chart, scatter chart

Description automatically generated**

Figure 7: The plot including the decision boundaries and the misclassified data points

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