**ENGR-421 HOMEWORK-2 REPORT**

Name-Surname: Barış KAPLAN

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

**import pandas as pd**

**import numpy as np**

**import math**

**import matplotlib.pyplot as plt**

-------------------------------------------------------------------------------------------------------------------------------

For reading the given csv files, I have utilized read\_csv function of the pandas library. To get rid of the headers in the csv files, I have assigned the header to None. I have read the csv files as follows:

**img\_file= 'hw02\_images.csv'**

**lbl\_file= 'hw02\_labels.csv'**

**imgD= pd.read\_csv(img\_file, header=None, delimiter=",")**

**lblD= pd.read\_csv(lbl\_file, header=None, delimiter=",")**

-------------------------------------------------------------------------------------------------------------------------------

For dividing the label data and clothing image data into training set and test set, I have utilized iloc[ ] function. I have divided the clothing image data and label data into training set and test set as follows:

**clothingImgNum=35000**

**train\_data\_of\_clothing\_images = imgD.iloc[0:30000]**

**test\_data\_of\_clothing\_images = imgD.iloc[30000:clothingImgNum]**

**train\_data\_of\_clothing\_image\_labels = lblD.iloc[0:30000]**

**test\_data\_of\_clothing\_image\_labels = lblD.iloc[30000:clothingImgNum]**

-------------------------------------------------------------------------------------------------------------------------------

For estimating the mean parameters, I have used apply() functions. For each of the 784 pixels in a clothing image, I have estimated the mean parameters. For estimating the standard deviation parameters, I have also used apply() functions. For each of the 784 pixels in a clothing image, I have estimated the standard deviation parameters.

I have repeated this process for all of the clothing image labels (labels= 1 for T-shirt, 2 for Dress, 3 for Coat, 4 for Shirt, and 5 for Bag. Total number of repetitions= 5). I have utilized np.std() function of numpy library to obtain the estimates of the standard deviation parameters for each of the 784 pixels in a clothing image. I have defined and utilized a function called my\_custom\_avr\_func to obtain the estimates of the mean parameters for each of the 784 pixels in a clothing image. You can find the estimations of the mean parameters in Figure 1, and the standard deviation parameters in Figure 2.

The purpose of the function called my\_custom\_avr\_func is to find the average of a list.

Table

Description automatically generatedTable

Description automatically generated

Figure 2: The estimations of the standard deviation parameters for all of the clothing image labels

Figure 1: The estimations of the mean parameters for all of the clothing image labels

For estimating the prior probability parameters, initially, I have calculated the length of the part of the training data set (training data set of the clothing image labels) where the labels are equal to 1. I have repeated this process for the other labels (other labels= 2, 3, 4, and 5). Then, to obtain the prior probability estimations, I have divided these subset lengths with the lengths of the training data set of the clothing image labels. You can find the found estimations of the prior probability parameters in the Figure 3.

A picture containing table

Description automatically generated

,

Figure 3: The estimations of prior probabilities for all of the clothing image labels

For finding the score values of the classes, I have utilized the formula in Figure 4.

Text, whiteboard

Description automatically generated

Figure 4: The formula I used for calculating the scores

Then, depending on the score values of the classes, I have selected a class and classified the data. By using the apply() function, to obtain the predictions, I have choosed a class for each image in the training set called “train\_data\_of\_clothing\_images” and in the test set called “test\_data\_of\_clothing\_images”.

By using the pd.crosstab function of the pandas library, I have displayed the confusion matrix for the training set and the confusion matrix for the test set. You can see the confusion matrix I found for the training set in Figure 5 and the confusion matrix I found for the test set in Figure 6.

I have written the following to display the confusion matrices:

**conf\_mat1 = pd.crosstab(trnPrd,lbls,rownames = ['y\_pred'],colnames = ['y\_truth'])**

**print(conf\_mat1)**

**conf\_mat2 = pd.crosstab(tstPrd,tstLbls,rownames = ['y\_pred'],colnames = ['y\_truth'])**

**print(conf\_mat2)**

A picture containing table

Description automatically generatedTable

Description automatically generated

Figure 6: The Confusion Matrix for the test set

Figure 5: The Confusion Matrix for the training set

We can observe from the confusion matrices that y\_pred and y\_truth are close for the class2 (There are a few misclassified points in the class2, approximately %5 of the data points are misclassified). For the class4, there are a lot of misclassified data points (for class4, approximately 90 percent of the data points are misclassified). Hence, the accuracy is not good for the class4. For the class1, approximately 40 percent of the data points are misclassified. This is better than the results of class4. For class3, approximately %22 of the data points are misclassified. This is better than the results of the class1. For class5, approximately %28 of the data points are misclassified. This is better than the results of the class1 but worse than the results of the class5.

**Misclassification Percentages**

Class1: %40

Class2: %5

Class3: %22

Class4: %90

Class5: %28

The accuracy order of the results for the classes (from low accurate to high accurate):

**class4<class1<class5<class3<class2**