Density Estimation and Classification Project Report

Name: Sudha Rani Ramisetti | **ID**: 1227288129

Analysis:

This project is about implementing a two-class Naïve Bayes classifier for digit classification (digit '0' and digit '1') on a subset of the MNIST dataset containing images of digits '0' and '1' by performing four major tasks: feature extraction, parameter calculation, implementation of Naïve Bayes classifiers and predictions of labels, and finally, calculating the accuracy of the predictions.

Training datasets and testing datasets are provided for digit0 and digit1. Feature extraction and parameter calculations have to be performed on training datasets. By using these parameters, predictions should be made about testing datasets.

Task1: Feature Extraction

In this task, I extracted two features from training datasets by converting the original data into 1x2 array, where each column represents a feature.

Feature1: Average brightness - the mean of each image pixel values in the training data.
Feature2: Standard brightness - the standard deviation of each image pixel values in the training data.

```
def extractFeatures(images):
    flatImages = images.reshape(images.shape[0], -1)
    avgBrightness = numpy.mean(flatImages, axis=1)
    stdBrightness = numpy.std(flatImages, axis=1)

    return avgBrightness, stdBrightness
```

```
avgBrightness0, stdBrightness0 = extractFeatures(train0)
avgBrightness1, stdBrightness1 = extractFeatures(train1)
```

Task2: Parameter calculation

In this task, I calculated the mean and variance of both features for train0, train1 datasets.

```
meanF1T0 = numpy.mean(avgBrightness0)
varianceF1T0 = numpy.var(avgBrightness0)

meanF2T0 = numpy.mean(stdBrightness0)
varianceF2T0 = numpy.var(stdBrightness0)

meanF1T1 = numpy.mean(avgBrightness1)
varianceF1T1 = numpy.var(avgBrightness1)

meanF2T1 = numpy.mean(stdBrightness1)
varianceF2T1 = numpy.var(stdBrightness1)
```

Task3: Implementation of Naïve Bayes Classifier

I used the parameters obtained in task2, to implement the classifier and to predict the labels for testing data sets (test0, test1).

I followed the below steps:

- First, I extracted features from testing data sets like in step1.
- Then I calculated the NB likelihood which is a product of Gaussian PDF of feature1 and feature2 for digit0 and digit1 for each data point in test data.

```
likelihood0 = calculatePDF(feature1[i], mean1F1, var1F1) * calculatePDF(feature2[i], mean1F2, var1F2)
likelihood1 = calculatePDF(feature1[i], mean2F1, var2F1) * calculatePDF(feature2[i], mean2F2, var2F2)
```

- Then I compared these two likelihood values to predict the label

```
predictedLabel = 0 if likelihood0 > likelihood1 else 1
```

- Gaussian PDF is calculated by using the formula:

$$\mathrm{PDF}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Where x=feature, μ=mean, σ^2=variance

```
normConstant = 1 /  numpy.sqrt(2 * numpy.pi * variance)
exponent = numpy.exp(-( (feature - mean) ** 2 / (2 * variance)))

return exponent * normConstant
```

Task4: Calculate Accuracy

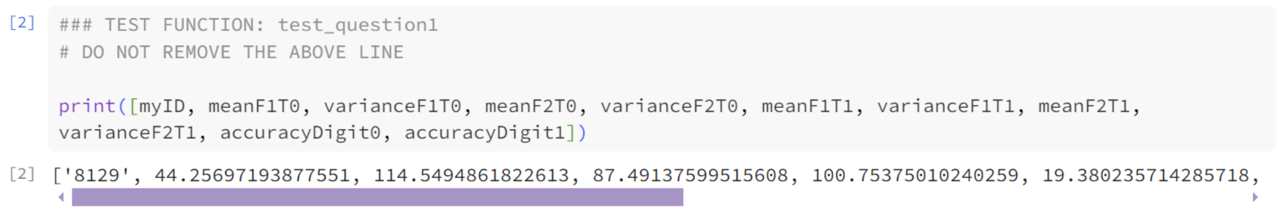
In this step, I calculated the accuracy of predicted labels in task3 for digit0 and digit1.

```
correctp0 = sum(p == 0 for p in p0)
correctp1 = sum(p == 1 for p in p1)

accuracyDigit0 = correctp0 / len(test0)
accuracyDigit1 = correctp1 / len(test1)
```

Output:

```
['8129', 44.25697193877551, 114.5494861822613, 87.49137599515608, 100.75375010240259,
19.380235714285718, 31.35577751623178, 61.37439414015311, 82.32411425530601, 0.9173469387755102,
0.9233480176211454]
```



```
[2]  ### TEST FUNCTION: test_question1
     # DO NOT REMOVE THE ABOVE LINE

     print([myID, meanF1T0, varianceF1T0, meanF2T0, varianceF2T0, meanF1T1, varianceF1T1, meanF2T1,
     varianceF2T1, accuracyDigit0, accuracyDigit1])

[2]  ['8129', 44.25697193877551, 114.5494861822613, 87.49137599515608, 100.75375010240259, 19.380235714285718,
```