Introduction:

I utilized the Jupyter notebook (lab) provided within Coursera platform inside week 3 for my coding. To make the code self-explanatory I included lots of comments to explain the codes. I then downloaded the notebook as a Python script, which I have included alongside this report.

There was a cell in the Jupyter notebook for importing the needed datasets. I edited the cell by doing the following:

1. Import more libraries that I will need for the assignment.
2. Replace myID (a 4 digit string) in the preexisting cell with the last 4 digit of my student ID – 1417
3. Edited and called the function "main" so that I can return the needed datasets (train0, train1, test0, test1).
4. Printed out the dimensions of the datasets to get a better picture of the datasets.

Task 1:

From the printed dimensions, I knew my datasets were three dimensional, in which the first dimensions vary (i.e. is the number of images) and the other two dimension (from pixel of each image) were 28. I generated 2-d data from the 3-d data by multiplying the rows and columns of the pixels each image. This resulted in 784 column datasets.

The second part of this task reduce the dimension of the datasets from 784 columns to two by taking the means (feature 1) and the standard deviation (feature 2). This resulted in two column representative datasets, with means in the first column and the standard deviations (std) in the second columns. I generated this representative datasets for both the test and training datasets of 0 and 1.

Task 2:

In task two I used only the training datasets. I generated the mean of feature1(mean) and feature2 (standard deviation) for digit 0 and digit 1. I also generated the variance of feature 1 and feature 2 for the two digits. I captured these results (means and variance) neatly in a pandas data frame (see table below), to ease my subsequent submission of my work in the graded project section.

meanFeature1_0	44.290374
varianceFeature1_0	5.697772
meanFeature2_0	88.673605
varianceFeature2_0	4.958382
meanFeature1_1	19.399015
varianceFeature1_1	4.211849
meanFeature2_1	62.158984
varianceFeature2_1	15.064590

Task 3:

<u>Bayes Theorem and Explanation of Naïve Bayes Classifier Implementation</u>

Bayes Theorem can be stated as:

Posterior = Likelihood * Prior / Evidence

Or

P(y|x) = P(x|y) * P(y) / P(x)

since P(x) is a constant,

P(y|x) is proportional to P(x|y) * P(y)

In this assignment a class of 0 or 1 is to be assigned based on two extracted features (mean and standard deviation of image pixels. The prior probabilities P(y=0) and P(y=0) are already assumed to be the same (0.5).

Usually to create a classifier model, the probability of given set of inputs for all possible values of the class variable y is found and the output with maximum probability is selected. This can be expressed mathematically as:

y[hart] = argmax{P(x|y) * P(y)}

Since the prior for both classes (0 and 1) are same, then it can simply be ignored, therefore:

y[hart] = argmax{P(x|y)}

<u>Classifier Implementation</u>

I created the following functions to do my prediction:

1. A function called "calculate_prob_ygivenX" which calculate the probability of a row (x1 and x2, i.e. the features 1 and 2 respectively) from the test dataset to be a digit given the mean of the mean, the mean of the std, the variance of the mean and the variance of the std of the respective feature in the training datasets.
2. A function called "argmax_y" which calls "calculate_prob_ygivenX" twice to calculate the probability of a row (pair of values – i.e. features 1 and 2) being digit 0 [the first call] and digit 1 [the second call]; using the means and variances that were calculated in task 2 above. "argmax_y" then use numpy.argmax to return the index of the maximum of the probabilities. Since the probability of digit 0 is the first index (0 in Python), if 0 is the maximum of the two probabilities, then 0 will be returned. Similarly if the probability of 1 is the maximum, 1 (the second index) is returned. This way argmax_y classify a row of a test dataset either as digit 0 or digit 1.
3. A function called "testclass_y" which predicts the labels for a dataset by iterating through each row of the dataset and calling "argmax_y" to classify each row.

Prediction of labels

To predict the labels for digit 0 and digit 1 test datasets, I called the function "testclass_y" with the argument being their respective two column datasets that were generated in task 1 above.

Task 4:

In this task, I simply calculated the accuracy by dividing the number of correctly predicted labels for a test dataset by the total number of labels in that test datasets.

Task 5:

I decided to check if my result that I coded from scratch with Python is right by cross checking with sklearn; since Naïve Bayes should produce the same result regardless. I repeated the analysis using the two dimensional datasets that had 784 columns. Please see the code under the section "TASK 5". Sklearn upheld my accuracy for digit 1 dataset to be 1.0. However, Sklearn disagreed with my result for digit 0 – 0.4020408163265306 instead of 0.5908163265306122 from scratch.

After trouble shooting, I realized that there was nothing wrong with my initial result. The problem is that argmax picks the first instance of the highest probability. So what happened is that coincidentally, substantial amount of the digit 0 datasets had equal probability for class 0 and class 1. If the probability for class 0 is placed before 1, you get the higher accuracy; but if the probabilities are switched you get the lower accuracy because class 1 is now favored over class 0.

Since I now have to choose one of the accuracies, I decided to use the high accuracy I originally got. This was done after consulting with a TSA (Anoop) during the live event on Friday (April 3rd), concerning this discrepancies. Anoop asked me to explain this in my report, so that it can be put into consideration during grading. That is why I have included it here. I have also included in “TASK 5” of the code, the alternative code to get the answer as produced by Sklearn. In this alternative code, I favored class 1 over 0 and got the approximately the same lower accuracy - 0.4091836734693878 from scratch.

References:

1. Lecture videos
2. https://www.geeksforgeeks.org/naive-bayes-classifiers/
3. https://scikit-learn.org/stable/modules/naive_bayes.html