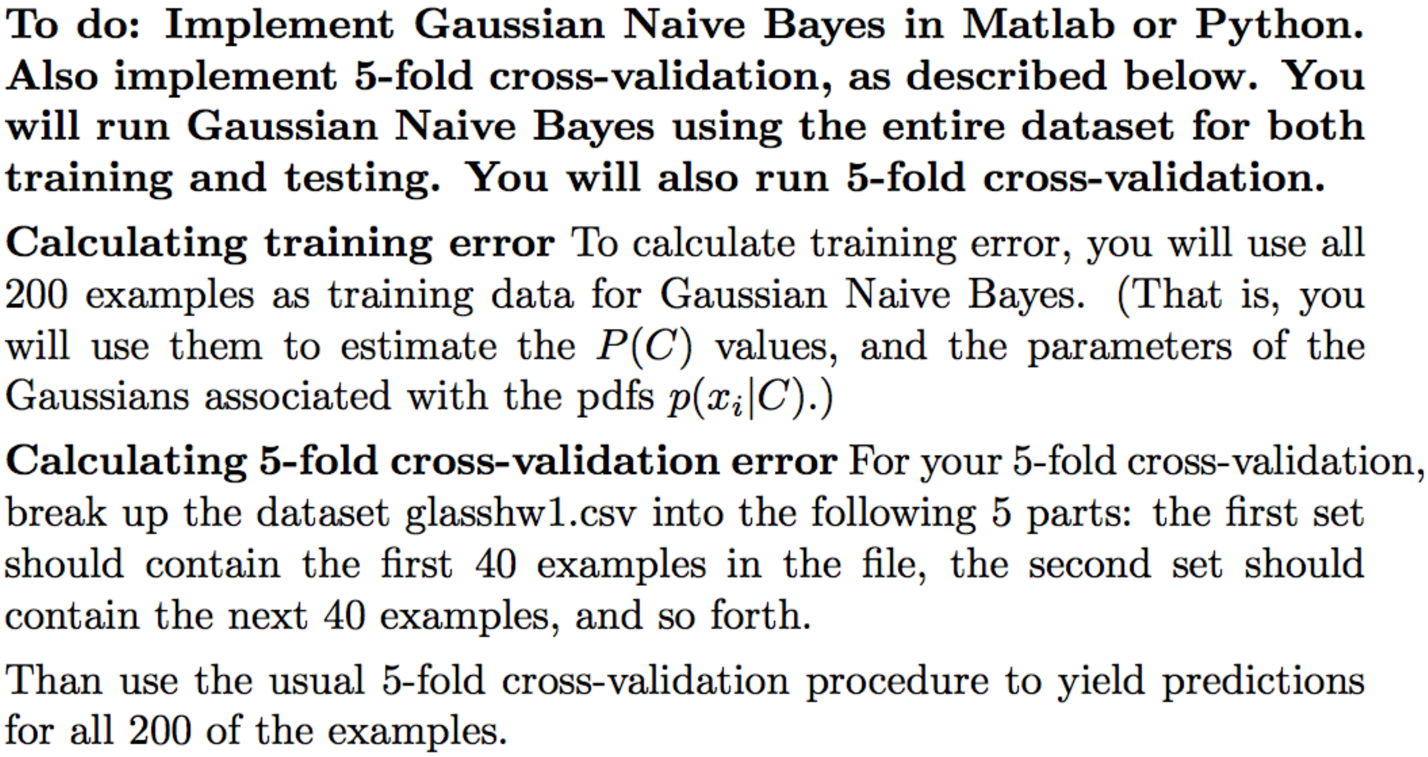

To do: Implement Gaussian Naive Bayes in Matlab or Python. Also implement 5-fold cross-validation, as described below. You will run Gaussian Naive Bayes using the entire dataset for both training and testing. You will also run 5-fold cross-validation.

Calculating training error To calculate training error, you will use all 200 examples as training data for Gaussian Naive Bayes. (That is, you will use them to estimate the $P(C)$ values, and the parameters of the Gaussians associated with the pdfs $p(x_i|C)$.)

Calculating 5-fold cross-validation error For your 5-fold cross-validation, break up the dataset glasshw1.csv into the following 5 parts: the first set should contain the first 40 examples in the file, the second set should contain the next 40 examples, and so forth.

Than use the usual 5-fold cross-validation procedure to yield predictions for all 200 of the examples.

Training details:

Estimation of $P(C)$: To estimate the $P(C)$ values, use frequency estimates. For example, to estimate $P(C = 1)$ calculate:
(number of examples in Class 1)/(total number of examples).

Estimation of $p(x_i|C)$: To estimate the mean of the Gaussian, use the average of the numbers as the estimate $\hat{\mu}$ of the mean.

$$\hat{\mu} = \frac{\sum_t x_t}{N}$$

where N is the number of examples involved.

For your estimate of the variance, use the following formula:

$$\hat{\sigma}^2 = \frac{\sum_t (x_t - \hat{\mu})^2}{N - 1}$$

Note that $N - 1$ is in the denominator here, not N.

Testing details: Multiplying lots of probabilities can lead to underflow. To avoid that, in deciding how to classify an example $x = (x_1, \ldots, x_9)$, you should find the value of $\log[p(x|C) * P(C)] = \log P(C) + \sum_{i=1}^{9} \log p(x_i|C)$ for each class C. (Use the natural log, ln.) Then classify the example according to the class achieving the maximum value for this expression. If there is a tie, classify the example according to the higher numbered class.

- During the run that uses the entire file for training and testing:
  - The estimated value of $P(C)$ for each class C.
  - The estimates $(\hat{\mu}, \hat{\sigma}^2)$ for the Gaussians corresponding to $p(x_i|C)$, for each attribute x_i and each class C_i. (so you need to output 18 pairs $(\hat{\mu}, \widehat{\sigma^2})$).
  - The prediction made on each of the 200 examples
  - The percentage error on the 200 examples. (This is the *training error*.)
- For the 5-fold cross validation run:
  - The prediction made on each of the 200 examples
  - The percentage error on the 200 examples. (This is the 5-fold cross-validation error.)

5. (a) For the run that used all 200 examples for both training and testing:

    i. What was the estimated value of $P(C)$ for $C = 1$?

    ii. What was the estimated value of $P(C)$ for $C = 2$?

    iii. What were the estimated values for $(\hat{\mu}, \widehat{\sigma^2})$ for the Gaussians corresponding to attribute Refractive Index and Class 1.

    iv. What were the estimated values for $(\hat{\mu}, \widehat{\sigma^2})$ for the Gaussians corresponding to attribute Calcium and Class 2.

    v. Which classes were predicted for the following examples: Examples 20, 60, 100, 140, and 180

    vi. What was the percentage training error?

   (b) For the run using 5-fold cross-validation,

    i. What was the percentage 5-fold cross-validation error?

    ii. Which classes were predicted for the following examples: Examples 20, 60, 100, 140, and 180

   (c) Sometimes a not-very-intelligent learning algorithm can achieve high accuracy on a particular learning task simply because the task is easy. To check for this, you can compare the performance of your algorithm to the performance of some very simple algorithms. One such algorithm just predicts the majority class (the class that is most frequent in the training set). This algorithm is sometimes called Zero-R. It can achieve high accuracy in a 2-class problem if the dataset is very imbalanced (i.e., if the fraction of examples in one class is much larger than the fraction of examples in the other). Run 5-fold cross-validation on your dataset, as before, but using Zero-R instead of Gaussian Naive Bayes. What accuracy is attained?