Due date: Apr. 3, 2020, 11:59 PM (Arlington time). You have **two** late days — use it at as you wish. Once you run out of this quota, the penalty for late submission will be applied. You can either use your late days quota (or let the penalty be applied). **Clearly indicate** in your submission if you seek to use the quota.

What to turn in:

1. Your submission should include your complete code base in an archive file (`zip, tar.gz`) and `q1/, q2/,` and so on), and a very very clear README describing how to run it.
2. A brief report (typed up, submit as a PDF file, NO handwritten scanned copies) describing what you solved and implemented and known failure cases. The report is **important** since we will be evaluating the grades mostly based on the report.
3. Submit your entire code and report to Canvas.

Notes from instructor:

- Start early!
- You may ask the TA or instructor for suggestions, and discuss the problem with others (minimally). But **all parts of the submitted code must be your own**.
- Use Matlab or Python for your implementation.
- Make sure that the TA can easily run the code by plugging in our test data.

Problem 1

(Non-parameteric density estimation **35pts**)

1. **(20pts)** Write a function `[p, x] = mykde(X,h)` that performs kernel density estimation on data X with bandwidth h. It should return the estimated density $p(x)$ and its domain x where you estimated the $p(x)$ for X in 1-D and 2-D.
   - Generate $N = 1000$ Gaussian random data with $\mu_1 = 5$ and $\sigma_1 = 1$. Test your function `mykde` on this data with $h = \{0.1, 1, 5, 10\}$. In your report, report the histogram of X along with the figures of estimated densities.
   - Generate $N = 1000$ 1-D Gaussian random data with $\mu_1 = 5$ and $\sigma_1 = 1$ and another Gaussian random data with $\mu_2 = 0$ and $\sigma_2 = 0.2$. Test your function `mykde` on this data with $h = \{0.1, 1, 5, 10\}$. In your report, report the histogram of X along with the figures of estimated densities.
2. **(15pts)** Generate 2 sets of 2-D Gaussian random data with $N_1 = 500$ and $N_2 = 500$ using the following parameters:

$$\mu_1 = [1, 0],\ \mu_2 = [0, 2.5],\ \Sigma_1 = \begin{bmatrix} 0.9 & 0.4 \\ 0.4 & 0.9 \end{bmatrix},\ \Sigma_2 = \begin{bmatrix} 0.9 & 0.4 \\ 0.4 & 0.9 \end{bmatrix}. \tag{1}$$

   Test your function `mykde` on this data with $h = \{0.1, 1, 5, 10\}$. In your report, report figures of estimated densities.

Problem 2

(Naive Bayes, **65pts**)

Generate 1000 training instances in two different classes (500 in each) from multi-variate normal distribution using the following parameters for each class

$$\mu_1 = [1, 0],\ \mu_2 = [0, 1],\ \Sigma_1 = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix},\ \Sigma_2 = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix} \tag{2}$$

and label them 0 and 1. Then, generate testing data in the same manner with 500 instances for each class, i.e., 1000 in total.

1. (**30pt**) Implement your Naive Bayes Classifier `[pred, posterior, err] = myNB(X,Y,X_test,Y_test)` whose inputs are the training data `X`, labels `Y` for `X`, testing data `X_test` and labels `Y_test` for `X_test` and returns predicted labels `pred`, posterior probability `posterior` with which the prediction was made and error rate `err`. Assume Gaussian (normal) distribution on the data: there are two parameters that realizes the probability density function (pdf), i.e., μ and σ. You can use functions such as `normpdf` or `pdf` in matlab (or equivalent functions in Python) to obtain likelihood from Gaussian pdf.

   For the experiments below, you should perform the experiments several times (e.g., 10 times) to find out meaningful performance (e.g., take an average) since the data generated are random every time. Derivation of Naive Bayes looks complicated, but its actual implementation should be simple if you understand the concept of Naive Bayes Classifier.

   - Perform prediction on the testing data with your code. In your report, report the accuracy, precision and recall as well as a confusion matrix. Also, make sure to include a scatter plot of data points whose labels are color coded (i.e., the samples in the same class should have the same color) in the report.
   - In your training data, change the number of examples in each class to $\{10, 20, 50, 100, 300, 500\}$ and perform prediction on the testing data with your code. In your report, show a plot of changes of accuracies w.r.t. the number of examples and write your brief obervation.
   - Now, in your training data, change the number of examples in class 0 as 700 and the other as 300. Perform prediction on the testing dataset. How does the accuracy change? Why is it changing? Write your own observation.

2. (**15pt**) Write a code to plot an ROC curve and calculate Area Under the Curve (AUC) based on the posterior for class 1 (i.e., the confidence measure for class 1 is the posterior). The implementation should be done on your own without using explicit library that lets you draw the curve. Report the ROC curves from the two cases discussed in above (i.e., 1) equal number of samples for each class and 2) unequal number of samples in the training data).

3. (**20pt**) Download Amazon reviews dataset with labels (Dataset: `https://www.kaggle.com/noushad24/amazon-reviews/download`). Use tf-idf weight matrix (You've done it in HW1) as features and perform 5-fold cross-validation with the Naive Bayes classifier. In your report, report the result (average accuracy, precision and reacall across all folds) and your observation.