Assignments (may use Julia or Python):

- Students are encouraged to use Jupyter or Pluto notebooks for their assignments & projects.
- All assignments must be received by 1159PM on the scheduled date.
- All assignments carry equal credit, adding up to 30% of the course grade.
- Each submission must be a single pdf document that includes **code** and **a report** (English sentences) summarizing the results.
- You may use any packages, code, or tools available on the internet, as long as you clearly cite the source.
- If you encounter programming difficulties, you may consult anyone for help, or study good code written by experts outside the class, so that you improve your programming skills.
- You may discuss a homework orally with other students if you work on different datasets. But the actual work on each homework must be done independently. Don't even look at other students' code for any assignment!
- The goal of ML with these assignments is to get the best possible accuracy on the training set for a 2-class problem, with <10% training errors with each class, but penalizing one kind of error (for the minority class) three times more than another. If *b1* and *b2* are the number of data points of the two classes, with *b1<b2,* of which *a1* and *a2* are the numbers of misclassified training data, respectively for the two classes, the ML algorithm must minimize

  $$q = (3a1+a2)/(3b1+b2) + 10\,(\max(0, a1/b1 - 0.1) + \max(0, a2/b2 - 0.1)).$$

  For example, if $a1=20, a2=10, b1=100, b2=200,$ then $q=70/500+10(0.2-0.1 + 0) = 1.14,$ whereas if $a1=10, a2=20, b1=100, b2=200,$ then $q=50/500+10(0+0) = 0.1,$ although the accuracy is the same in both cases; hence the latter solution is preferable.

Tentative list of assignments:

- HW0 (due on 2/4/25):
  - Select a 2-class classification problem with at least ten input attributes or features and at least 1000 labeled data points, using any dataset publicly available on the internet (e.g., UCI, Stanford, Kaggle).
  - Keep aside 100 data points from each class, to be used for testing.
  - Unbalance the remaining (training) data, so that you have three times as many points from one of the classes, (e.g., 600 + 200 data points); you can do this by deleting some points from one class, or by duplicating some points and adding a little noise.
  - Randomly relabel 5% of the training data from each class, e.g., labeling 5% of the dogs as cats, and vice versa. This ensures that very high accuracy is not achievable.
  - Train a shallow feedforward neural network (with sigmoidal node functions and one hidden layer with twice as many nodes as the input dimensionality, e.g., 12-24-1 architecture if there are 12 input attributes) using backpropagation (possibly with ADAM optimizer), while keeping track of performance on test data during the training process. You can use some function other than MSE as the loss function for backpropagation.
  - Repeat the experiment ten times, each time starting with a different set of randomly initialized weights.
  - Summarize the results **using two curves in one graph,** plotting the average ***q*** (on the y-axis) against **log*(number of weight updates)***, for training data and test data.
  - Also show the confusion matrices (separately) for training data and test data.