Why?

The purpose of this project is to explore some techniques in supervised learning. It is important to realize that understanding an algorithm or technique requires understanding how it behaves under a variety of circumstances. As such, you will be asked to "implement" some simple learning algorithms (for sufficiently small values of implement, meaning I don't really want you to implement anything at all), and to compare their performance. The latter is quite involved and also quite possibly different from what you are used to; however, it is central and in many ways the whole point.

The Means

In this assignment you will go through the process of exploring your chosen datasets, tuning the algorithms you learned about, and writing a thorough analysis of your findings. All that matters is the analysis. It doesn't matter if you implement any learning algorithms yourself (you probably shouldn't) as long as you participate in this journey of exploring, tuning, and analyzing. Concretely, this means you may program in any language that you wish, and you are allowed to use any library you wish **as long as it was not written specifically to solve this assignment,** and as long as a TA can recreate your experiments on a standard linux machine if necessary (we know how to install a pip package). You might want to look at scikit-learn or Weka, for example.

Some **examples** of acceptable libraries:

- Machine learning algorithms: scikit-learn (python), Weka (java), e1071/nnet/random forest(R), ML toolbox (matlab), tensorflow/pytorch (python)
- Scientific computing: numpy/scipy(python), Matlab, R
- Plotting: matplotlib (python), seaborn (python), Matlab, R

You can use other libraries as long as they fulfill the conditions above. If you are unsure, ask a TA (but please use common sense first)! There is no trick here for you to overthink. Again, the key issue is that I don't care that you implement any of the learning algorithms below; however, I care very much about your analysis.

The Problems Given to You

You should implement five learning algorithms. They are:

- Decision trees with some form of pruning
- Neural networks
- Boosting
- Support Vector Machines
- *k*-nearest neighbors

Each algorithm is described in detail in your textbook, the handouts, and all over the web. In fact, instead of implementing the algorithms yourself, you may (and by may I mean should) use software packages that you find elsewhere; however, if you do so you should provide proper attribution. Also, you will note that you have to do some fiddling to get good results, graphs and such, so even if you use another's package, you may need to be able to modify it in various ways.

Decision Trees. For the decision tree, you should implement or steal a decision tree algorithm (and by "implement or steal" I mean "steal"). Be sure to use some form of pruning. You are not required to use information gain (for example, there is something called the GINI index that is sometimes used) to split attributes, but you should describe whatever it is that you do use.

Neural Networks. For the neural network you should implement or steal your favorite kind of network and training algorithm. You may use networks of nodes with as many layers as you like and any activation function you see fit.

Boosting. Implement or steal a boosted version of your decision trees. As before, you will want to use some form of pruning, but presumably because you're using boosting you can afford to be much more aggressive about your pruning.

Support Vector Machines. You should implement (for sufficiently loose definitions of implement including "download") SVMs. This should be done in such a way that you can swap out kernel functions. I'd like to see at least two.

***k*-Nearest Neighbors.** You should "implement" (the quotes mean I don't mean it: steal the code) *k*NN. Use different values of *k*.

Testing. In addition to implementing (wink) the algorithms described above, you should design two interesting classification problems. For the purposes of this assignment, a classification problem is just a set of training examples and a set of test examples. I don't care where you get the data. You can download some, take some from your own research, or make some up on your own. Be careful about the data you choose, though. You'll have to explain why they are interesting, use them in later assignments, and come to really care about them.

What to Turn In

You must submit:

1. a file named *README.txt* containing instructions for running your code (see note below)
2. a file named *yourgtaccount-analysis.pdf* containing your writeup (GT account is what you log in with, not your all-digits ID)

Note below: we need to be able to get to your code and your data. Given what you're turning in, you will want to arrange for an URL of some sort. Having said that, providing

all of Weka isn't necessary, for example, because I can get it myself; however, you should at least provide any files you found necessary to change and enough support and explanation so we could reproduce your results if we really wanted to do so. In any case, include all the information in *README.txt*

The file *yourgtaccount-analysis.pdf* should contain:

- a description of your classification problems, and why you feel that they are interesting. Think hard about this. To be at all interesting the problems should be non-trivial on the one hand, but capable of admitting comparisons and analysis of the various algorithms on the other. Avoid the mistake of working on the largest most complicated and messy dataset you can find. The key is to be interesting and clear, no points for hairy and complex.
- the training and testing error rates you obtained running the various learning algorithms on your problems. At the very least you should include graphs that show performance on both training and test data as a function of training size (note that this implies that you need to design a classification problem that has more than a trivial amount of data) and--for the algorithms that are iterative--training times/iterations. Both of these kinds of graphs are referred to as learning curves, BTW.
- analyses of your results. Why did you get the results you did? Compare and contrast the different algorithms. What sort of changes might you make to each of those algorithms to improve performance? How fast were they in terms of wall clock time? Iterations? Would cross validation help (and if it would, why didn't you implement it?)? How much performance was due to the problems you chose? How about the values you choose for learning rates, stopping criteria, pruning methods, and so forth (and why doesn't your analysis show results for the different values you chose? Please do look at more than one. And please make sure you understand it, it only counts if the results are meaningful)? Which algorithm performed best? How do you define best? Be creative and think of as many questions you can, and as many answers as you can but a lot of the questions boil down to: why... WHY WHY WHY?

For the sanity of your graders, please keep your analysis as short as possible while still covering the requirements of the assignment: to facilitate this sanity, **analysis writeup is limited to 12 pages**.

Scoring Criteria

You are being scored on your analysis more than anything else. Roughly speaking, implementing everything and getting it to run is worth maybe 0% of the points on this assignment (I know you don't believe me, but in fact, steal the code; I not only don't care, I am encouraging you to use one of the many packages available both from the resources

page and on the web). Of course, analysis without proof of working code makes the analysis suspect.

The key thing is that your explanations should be both thorough and concise. Imagine you are writing a paper for the major conference in your field the year you will be graduating and you need to impress all those folks who will be deciding whether to interview you later. You don't want them to think you're shallow do you? Or that you're incapable of coming up with interesting classification problems, right? And you surely don't want them to think that you make up for a lack of content by blathering on about irrelevant aspects of your work? Of course not.

Let me point out here that for most of you (including the you reading this right now, I'm talking to you not just everyone else in the course but you) this form of assignment is not something you've done a lot. It is not about test cases and clearcut answers; rather, it is about understanding your results and articulating that understanding. I strongly recommend that you attend or listen to office hours for this and future assignments (actually because you've read the syllabus / front page, you know that watching office hours is required). Like machine learning itself, the assignments are iterative and require back-and-forth engagement to fully grok. Let us grok together.

Finally, I'd like to point out that I am very particular about the format of the assignments. Follow the directions carefully. Failure to turn in files with the proper naming scheme, or anything else that makes the graders' lives unduly hard is simply going to lead to an ignored assignment. I am remarkably inflexible about this. Also, there will be no late assignments accepted, so start now. Have fun. One day you'll look back on this and smile. There may be tears, but they will be tears of joy.

Rescoring Criteria

When your assignment is scored, you will receive feedback explaining your errors (and your successes!) in some level of detail. This feedback is for your benefit, both on this assignment and for future assignments. It is considered a part of your learning goals to internalize this feedback.

If you are convinced that your score is in error in light of the feedback, you may request a rescore within a week of the score and feedback being returned to you. A rescore request is only valid if it includes an explanation of where the grader made an error. There will be an Ed Discussion post detailing the rescoring process.

It is important to note that because we consider your ability to internalize feedback a learning goal, we also assess it. This ability is considered 10% of each assignment. We default to assigning you full credit. If you request a rescore and do not receive at least 5 points as a result of the request, you will lose those 10 points.

A note about plagiarism and proper citations

Probably the easiest way to fail this class is to plagiarize. Read the note on Ed when it arrives and make sure it doesn't happen.

BTW, I consider using the code of others in this class to perform the analysis itself to be plagiarism. Way above at the beginning of this assignment I note that I do not care about your implementing machine learning algorithms (and I mean it: I do not believe the learning value is worth it for this course); however, I do care very much that you understand why your algorithms work and how they are affected by your choices in data and hyper parameters (I believe very, very much in the learning value of this process).

The assignments are actually designed to force you to immerse yourself in the empirical and engineering side of ML that one must master to be a viable practitioner and researcher. The phrase "as long as you participate in this journey of exploring, tuning, and analyzing" is the key one here. Taking someone else's random seeds, hyper-parameters, and such is, in fact, avoiding the work I care about and circumvents this process because they are the results of the process. Do not circumvent the process.