HOMEWORK 2

>>NAME HERE<<
>>WISC ID HERE<<

Instructions: You can choose any programming language, as long as you implement the algorithm from scratch (e.g., do not use sklearn or any other library's decision tree implementation on Questions 1 to 7). Use this latex file as a template to develop your homework. Submit your homework on time as a single zip (containing the pdf and the code) file to Canvas. Please check Piazza for updates about the homework. For each question below, let θ be the last digit of your WISC-ID. It is recommended that you use Python for your solutions.

Setup: a Simple Decision Tree

You are to implement a decision-tree learner for classification. To simplify your work, this will not be a general purpose decision tree. Instead, your program can assume that

- each item has two continuous features $\mathbf{x} \in \mathbb{R}^2$
- the class label is binary and encoded as $y \in \{0, 1\}$
- data files are in plain-text with one labeled item per line, separated by white-space:

$$\begin{array}{cc} \mathbf{x}_1^{(1)} & \mathbf{x}_2^{(2)} \quad y^{(1)} \\ \mathbf{x}_1^{(2)} & \mathbf{x}_2^{(2)} \quad y^{(2)} \\ & \cdots \\ \mathbf{x}_1^{(n)} & \mathbf{x}_2^{(n)} \quad y^{(n)} \end{array}$$

Your program should implement a decision tree learner according to the following guidelines:

- Candidate splits (j, c) for numeric features should use a threshold c in feature dimension j in the form of $\mathbf{x}_j^{(i)} \geq c$, for some i.
- c should be on values of that dimension present in the training data; i.e., the threshold is on training points, not in between training points. You may enumerate all features, and for each feature, use all possible values for that dimension.
- You may skip those candidate splits with zero split information (i.e., the entropy of the split), and continue the enumeration.
- The left branch of such a split is the "then" branch, and the right branch is "else".
- Splits should be chosen using information gain ratio. If there is a tie you may break it arbitrarily.
- The stopping criteria (for making a node into a leaf) are that
  - the node is empty, or
  - the entropy of any candidate split is zero, or
  - all splits have zero gain ratio (if the entropy of the split is non-zero)
- To simplify, whenever there is no majority class in a leaf, let it predict $y = 1$.

1 Questions

1. (Our algorithm stops at pure labels) [5 pts] If a node is not empty but contains training items with the same label, why is it guaranteed to become a leaf? Explain. You may assume that the feature values of these items are not all the same.

2. (Our algorithm is greedy) [5 pts] Handcraft a small training set where both classes are present but the algorithm refuses to split; instead it makes the root a leaf and stops. Importantly, if we were to manually force a split, the algorithm will happily continue splitting the data set further and produce a deeper tree with zero training error. You should (1) plot your training set, (2) explain why. (Hint: you don't need more than a handful of items.)

3. (Gain ratio exercise) [10 pts] Use the training set `Druns.txt`. For the root node, list all candidate cuts and their information gain ratio. If the entropy of the candidate split is zero, please list its mutual information (i.e., information gain).

   (Hint: to get $\log_2(x)$ when your programming language may be using a different base, use $\log(x)/\log(2)$. Also, please follow the split rule in the first section.)

4. (The king of interpretability) [10 pts] Decision trees are not the most accurate classifiers in general. However, they are useful, largely due to their interpretability: a data scientist can easily explain a tree to a non-data scientist. Build a tree from `D3leaves.txt`. Then manually convert your tree to a set of logic rules. Show the tree [1] and the rules.

5. (Or is it?) [20 pts] For this question only, make sure you DO NOT VISUALIZE the datasets or plot your tree's decision boundary in the 2D $\mathbf{x}$ space. If your code does that, turn it off before proceeding. This is because you want to see your own reaction when trying to interpret a tree. You will get points no matter what your interpretation is. And we will ask you to visualize them in the next question anyway.

   - Build a decision tree on `D1.txt`. Show it to us in any format (e.g., could be a standard binary tree with nodes and arrows, and denote the rule at each leaf node; or as simple as plaintext output where each line represents a node with appropriate line number pointers to child nodes; whatever is convenient for you). Again, do not visualize the data set or the tree in the x input space. In real tasks, you will not be able to visualize the whole high dimensional input space anyway, so we don't want you to "cheat" here.
   - Look at your tree in the above format (remember, you should not visualize the 2D dataset or your tree's decision boundary) and try to interpret the decision boundary in human understandable English.
   - Build a decision tree on `D2.txt`. Show it to us.
   - Try to interpret your `D2` decision tree. Is it easy or possible to do so without visualization?

6. (Hypothesis space) [10 pts] For `D1.txt` and `D2.txt`, do the following separately:

   - Produce a scatter plot of the data set.
   - Visualize your decision tree's decision boundary (or decision region, or some other ways to clearly visualize how your decision tree will make decisions in the feature space). Then discuss why the size of your decision trees on `D1` and `D2` differ. Relate this to the hypothesis space of our decision tree algorithm.

7. (Learning curve) [20 pts] We provide a data set `Dbig.txt` with 10000 labeled items. Caution: `Dbig.txt` is sorted.

   - You will randomly split `Dbig.txt` into a candidate training set of 8192 items and a test set (the rest). Do this by generating a random permutation, and split at 8192. You should use as seed in the random generator, the last digit of your wisc-id (θ). A code for this task should be the following:

     ```
     import numpy as np
     np.random.RandomState(seed=Last-Digit).permutation(n)
     ```

[1] When we say show the tree, we mean either the standard computer science tree view, or some crude plain-text representation of the tree – as long as you explain the format. When we say visualize the tree, we mean a plot in the 2D x space that shows how the tree will classify any points.

- Generate a sequence of five nested training sets $D_{32} \subset D_{128} \subset D_{512} \subset D_{2048} \subset D_{8192}$ from the candidate training set. The subscript n in D_n denotes training set size. The easiest way is to take the first n items from the (same) permutation above. (Use the same seed as above). This sequence simulates the real world situation where you obtain more and more training data.
- For each D_n above, train a decision tree. Measure its test set error err_n. Show three things in your answer: (1) List n, number of nodes in that tree, err_n. (2) Plot n vs. err_n. This is known as a learning curve (a single plot). (3) Visualize your decision trees' decision boundary (five plots).

2 sklearn

[10 pts] Learn to use sklearn https://scikit-learn.org/stable/. Use sklearn.tree.DecisionTreeClassifier to produce trees for datasets $D_{32}, D_{128}, \ldots D_{8192}$. Show two things in your answer: (1) List n, number of nodes in that tree, err_n. (2) Plot n vs. err_n.

3 Lagrange Interpolation

[10 pts] Fix the interval $[0, 4\pi]$ and sample $n = 100$ points x from this interval (a) uniformly (b) by drawing samples according to a Gaussian distribution with mean $\mu = 2\pi - \theta\pi/10$ and standard deviation $\sigma = \pi/6$ and rejecting any sample outside the interval. Use these to build a training set consisting of n pairs (x, y) by setting function $y = \cos(x + \theta\pi/10)$, θ is again the last digit of your id.

Build a model f by using Lagrange interpolation, as discussed in lecture. Generate a test set using the same distribution as your test set. Compute and report the resulting model's train and test error. What do you observe? What do you observe if you increase σ to be $\pi/4$ and $\pi/2$?