HOMEWORK 2

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

Instructions: Although this is a programming homework, you only need to hand in a pdf answer file. There is no need to submit the latex source or any code. You can choose any programming language, as long as you implement the algorithm from scratch (e.g. do not use Weka on questions 1 to 7).
Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file to Canvas. Please check Piazza for updates about the homework.

1 A Simplified 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 plaintext with one labeled item per line, separated by whitespace:

$$x_{11} \quad x_{12} \quad y_1$$
$$\dots$$
$$x_{n1} \quad x_{n2} \quad y_n$$

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 $x_{\cdot j} \geq c$.
- 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.
- The left branch of such a split is the "then" branch, and the right branch is "else".
- Splits should be chosen using mutual information (i.e. information gain). 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
  - all splits have zero mutual information
- To simplify, whenever there is no majority class in a leaf, let it predict $y = 1$.

2 Questions

1. (Our algorithm stops at pure labels) [10 pts] If a node is not empty but contains training items with the same label, why is it guaranteed to become a leaf? Explain.
   The decision tree works by trying to maximize the information gain or minimizing the entropy. If a node is non empty but has training items has the same labels, the entropy is zero. Trying to create a split will give an information gain of zero. Since information gain is zero, the algorithm stops and all items are classified under a single label in that node. (Points are given for mentioning in any form that information gain is 0.)

2. (Our algorithm is greedy) [10 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 stop; 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. (Half points for plotting the data and half for the explanation. Any dataset is accepted as long as the explanation is correct and the dataset is legible.)
   The data points should be such that the information gain should be zero irrespective of the split. This suggests that the data should be such that for any root split, the both nodes should have equal number of items of both class. This would make the information gain zero, and the algorithm will stop at the root. An example of this data could simply be 4 points : (0,0,0),(1,0,1),(0,1,1),(1,1,0).(Any other data that gives an information gain of zero is okay.) If the splits are chosen as $x_1 \geq 0$ or $x_2 \geq 0$, the entropy would still be one. It remains 1 even if the splits chosen $x_1 \geq 1$ or $x_2 \geq 1$. Hence, the root becomes a leaf node. Now, if we force a split at say $x_1 \geq 1$, then the algorithm proceeds further with an information gain of 1 and ends with zero training error.

3. (Mutual information exercise) [10 pts] Use the training set Druns.txt. For the root node, list all candidate cuts and their mutual information. Hint: to get $\log_2(x)$ when your programming language may be using a different base, use `log(x)/log(2)`. Being a very small straightforward dataset, the candidate cuts and mutual information will be similar.

Dimension	Candidate Cut	Mutual Information
1	0.1	0.0441
1	0	0
2	-2	0
2	-1	0.0441
2	0	0.0382
2	1	0.0048
2	2	0.001
2	3	0.0163
2	4	0.0494
2	5	0.1051
2	6	0.1995
2	7	0.0382
2	8	0.189

Or:

	0.04418
	0.04418
	0.03827
$x1 \geq 0$:1	0.19959
$x2 \geq -1$:0	0.03827
$x2 \geq 0$:0	0.18905
$x2 \geq 6$:0	
$x2 \geq 7$:0	
$x2 \geq 8$:0	

4. (The king of interpretability) [10 pts] Decision tree is not the most accurate classifier in general. However, it persists. This is largely due to its rumored 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. Half the marks are for the tree and other half for explaining the tree. Different logic rules are allowed. If some kind of explanation is given and is correct but the logic rules are not given, 2 points will be deducted.

$$Y \Leftrightarrow (x1 \geq 10) \vee (x2 \geq 3)$$

Or Y <=> (x2>=2)v (x1>=10)

5. (Or is it?) [20 pts] For this question only, make sure you DO NOT VISUALIZE the data sets or plot your tree's decision boundary in the 2D **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. Points are as follows for each subquestion: (5,5,5,5)

   - 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 Weka style plaintext tree; 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.

[1]When we say show the tree, we mean either the standard computer science tree view, or some crude plaintext 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.

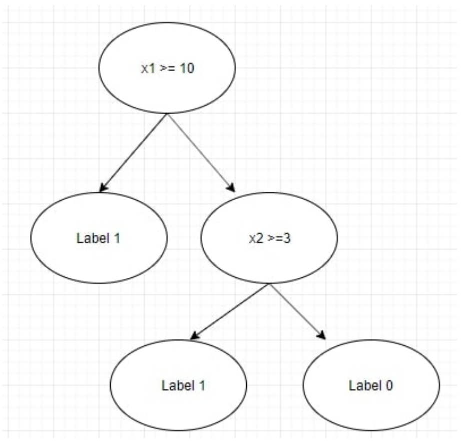


Figure 1: Tree for Q4

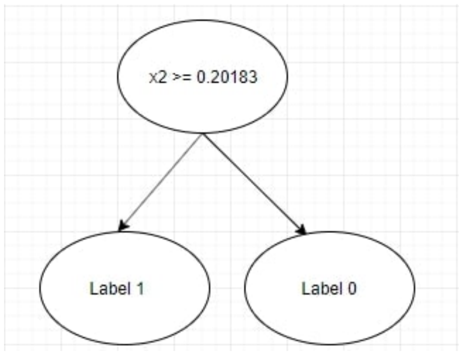


Figure 2: Q5

- 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. The tree can be interpreted to mean that all items labelled 1 are above a line $x_2 = 0.20183$ and all items labeled 0 are below this line. A straight line parallel to an axis separates the data.
- Build a decision tree on D2.txt. Show it to us. The tree is very complicated with around 55 nodes. Points are given if the tree is similar enough to the expected one. If the number of nodes is the same order as ours, full points are given. Otherwise if the tree is given and the depth is "off", some points will be taken off. (Points will be given as long as the tree is at least similar.)
- Try to interpret your D2 decision tree. This tree is very difficult to interpret without visualizing it in space.

6. (Hypothesis space) [10 pts] For D1.txt and D2.txt, do the following separately: Each subquestion has 5 points(2 for the scatter plot and 3 for the decision boudary).

   - 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). The boundary should be clearly visible and resemble those in the answers. If the boundary is shown and differs from the expected one, 1or 2 points will be taken off depending on how incorrect it is.

   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. The data in D1 is clearly separated by a line parallel to one of the axis. On the other hand, the data in D2 is separated by a slanting line which is not parallel to any axis. In the case of D2, one split along the line parallel to the axis is enough to split the data. To simulate the line parallel to the axis, the algorithm makes a lot of small splits along the axis. Since there are a lot of cuts, the tree is pretty deep.

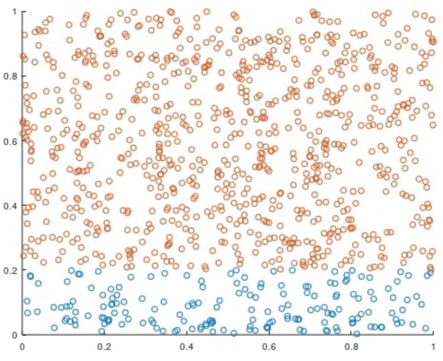
(11).jpg 

Figure 3: Q5

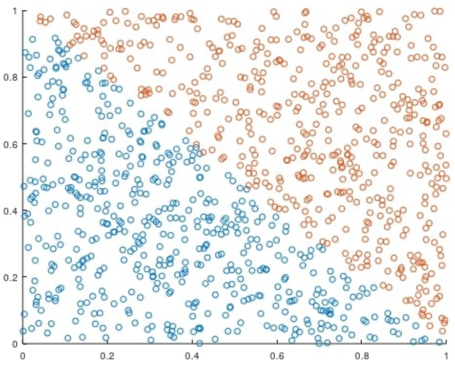
(12).jpg 

Figure 4: Q5

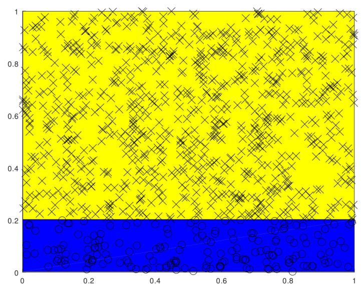


Figure 5: Q5

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.
   - 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

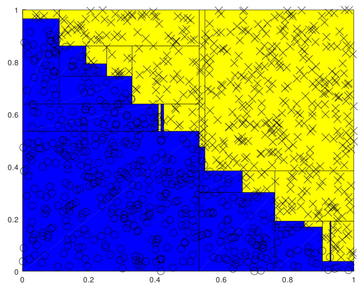
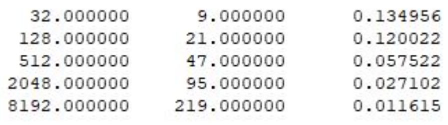


Figure 6: Q5

the first n items from the (same) permutation 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). 3 points for the (1), 2 points for (2) and 3 points each for each graph in (3). The number of nodes should be in the same order as the solution. A point will be taken off for unreasonable number of nodes.
  The error should be within 5% of the given numbers. Points will be deducted for large errors.
  The graph should resemble the one given.
  The decision boundaries should be approximately similar to the ones shown. Points will be deducted for a boundary which looks very different. Graphs for (3) are given on the last page.



```
  32.000000      9.000000      0.134956
 128.000000     21.000000      0.120022
 512.000000     47.000000      0.057522
2048.000000     95.000000      0.027102
8192.000000    219.000000      0.011615
```

Figure 7: Q7. n vs error

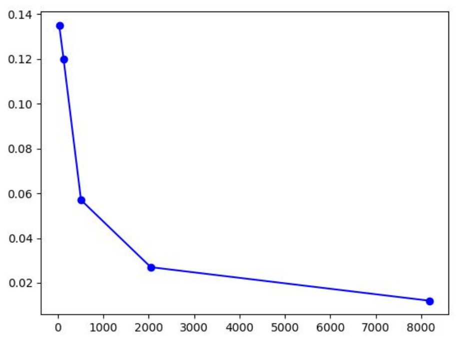


Figure 8: Q7 n vs error graph

3 Weka [10 pts]

Learn to use Weka https://www.cs.waikato.ac.nz/~ml/weka/index.html. Convert appropriate data files into ARFF format. Use trees/J48 as the classifier and default settings. Produce five Weka trees for $D_{32}, D_{128} \dots D_{8192}$. Show two things in your answer: (1) List n, number of nodes in that tree, err_n. (2) Plot n vs. err_n. (5 points for the table and 5 points for the graph.) The numbers just need to be in the same order as the ones given here. Exact answers are not expected.
The graph should resemble the one given.

n	Number of Nodes	Error
32	9	0.187
128	17	0.094
512	27	0.043
2048	59	0.026
8192	147	0.017

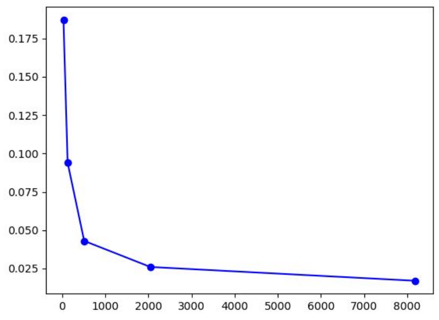


Figure 9: Weka n vs error

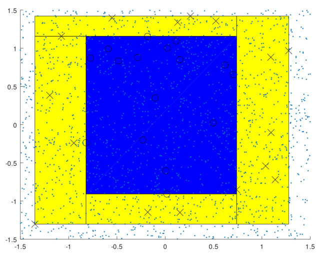


Figure 10: Q7 D32

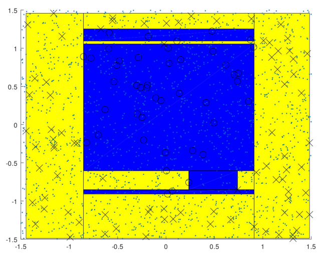


Figure 11: Q7 D128

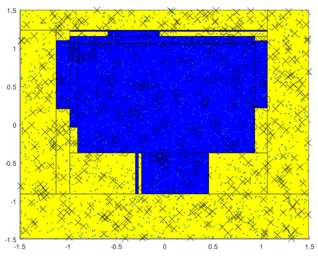


Figure 12: Q7 D512

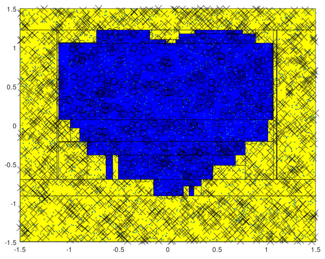


Figure 13: Q7 D2048

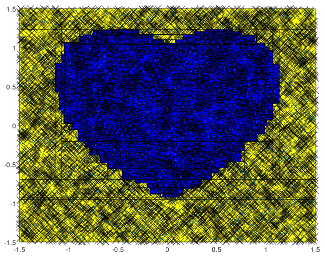


Figure 14: Q7 D8192