CS 6375- ASSIGNMENT 2

Please read the instructions below before starting the assignment.

- This assignment consists of written problems (Part I) and programming part (Part II). Please create separate folders named **parti** and **partii** and zip them together for submission.

- You should use a cover sheet, which can be downloaded at:
  http://www.utdallas.edu/~axn112530/cs6375/CS6375_CoverPage.docx

- You are allowed to work in pairs i.e. a group of two students is allowed. Please write the names of the group members on the cover page.

- You have a total of 4 free late days for the entire semester. You can use at most 2 days for any one assignment. After that, there will be a penalty of 10% for each late day. The submission for this assignment will be closed 2 days after the due date.

- **For part I:**
  - You can submit a scanned handwritten or typed solution. If handwritten, it should be legible and clear.
  - Show your calculations and steps clearly.

- **For part II:**
  - Before starting the assignment, please read about the ID3 algorithm at the resources below:
  https://en.wikipedia.org/wiki/ID3_algorithm
  http://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm
  Chapter 3 of Tom Mitchell's book (read page 56)
  Online version available at:
  http://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-dectrees.pdf

  - You are free to use any of the following programming languages for Part II -
  Java, Python, C++,, Ruby. You cannot use any packages or libraries.
  The instructions for compiling and running your code must be specified in the README file.

- **Your code will be checked for plagiarism. Do not copy code from online resources.**

CS 6375- ASSIGNMENT 2
DECISION TREE INDUCTION

Note:
1. Your solution to this assignment must be submitted via eLearning.
2. Whenever possible, you should provide brief justifications for your solution.
3. You may work in a group of two students or individually.

Part I: Written Problems (30 points)

1. Representing Boolean Functions (10 points)

Give decision trees to represent the following concepts. Your decision tree must contain as few nodes as possible. You can assume A, B, and C are Boolean variables.

(a) $Y = (\neg A \vee B) \wedge \neg(C \wedge A)$ where ¬ represents the NOT operator

(b) $Y = (A \oplus B) \wedge C$ where the symbol $\oplus$ represents the XOR logical operator

(c) $Y = (A \vee B) \wedge (B \vee C) \wedge (A \vee C)$

(d) $Y = (A \vee B) \wedge \neg A \wedge \neg B$ where ¬ represents the NOT operator

2. Decision Trees (20 points)

In this question, you will use the ID3 algorithm to create a decision tree for the dataset given below. There are three Boolean attributes X1, X2, and X3 and a Boolean class attribute.
Be sure to show detailed calculations for each step including entropy and information gain values. Draw a plot of the final tree that you obtain and show the class labels for the leaf nodes. Also indicate the set of instances that are associated with each leaf node.

Instance	X1	X2	X3	Class
1	1	0	0	1
2	0	1	0	1
3	0	0	0	0
4	1	0	1	0
5	0	0	0	0
6	1	1	0	1
7	0	1	1	0
8	1	0	0	1
9	0	0	0	0
10	1	0	0	1

Part II: Programming (70 points)

In this part, you will implement the ID3 decision tree learning algorithm using any one of the following programming languages - Java, Python, C++, Ruby. You cannot use any package or library for this assignment.

To simplify things, you can assume that the data used to test your implementation will contain only Boolean (0 or 1) attributes and Boolean (0 or 1) class values. You can assume that there will be no missing data or attributes. You can also assume that the first row of the dataset will contain column names and each non-blank line after that will contain a new data instance. Within these constraints, your program should be able to read and process any dataset containing any number of attributes. You can assume that the last column would contain the class labels.

A couple of datasets are provided. You have to build your model using the training dataset, check the model and prune it with the validation dataset, and test it using the testing dataset.

Below is a **summary of the requirements:**

- Build a binary decision tree classifier using the ID3 algorithm
- Your program should read four arguments from the command line – complete path of the training dataset, complete path of the validation dataset, complete path of the test dataset, and the pruning factor (explained later).
- The datasets can contain any number of Boolean attributes and one Boolean class label. The class label will always be the last column.
- The first row will define column names and every subsequent non-blank line will contain a data instance. If there is a blank line, your program should skip it.

- **Printing Decision Tree:**
  Your program should contain a print method that should output the current tree to the screen. It should be in the following format:

```
wesley = 0 :
| honor = 0 :
| | barclay = 0 :  1
| | barclay = 1 :  0
| honor = 1 :
| | tea = 0 :  0
| | tea = 1 :  1
wesley = 1 :  0
```

  The above tree shows that:
  if (wesley = 0 ^ honor = 0 ^ barclay = 0) then class = 1
  and also that:
  if (wesley = 0 ^ honor = 0 ^ barclay = 1) then class = 0
  and also that:
  if (wesley = 0 ^ honor = 1 ^ tea = 0) then class = 0
  and so on

- **Printing Summary and Results:**
  After reading all the data instances, you should output a summary of the datasets, and compute the pre-pruned accuracy on the training data and also accuracy of the model on the test dataset and output them to the screen. You should also output the plot of the decision tree model. For example,
  Pre-Pruned Accuracy

  Number of training instances = 100
  Number of training attributes = 5
  Total number of nodes in the tree = 20
  Number of leaf nodes in the tree = 8
  Accuracy of the model on the training dataset = 81.2%

  Number of validation instances = 50
  Number of validation attributes = 5
  Accuracy of the model on the validation dataset before pruning = 72.1%

  Number of testing instances = 20
  Number of testing attributes = 5
  Accuracy of the model on the testing dataset = 60.8%

- **Pruning of the Decision Tree**
  After the decision tree has been constructed, you will check the pruning factor, which will be given by the third argument to your program. The pruning factor is defined as the fraction of the nodes that you will prune. For example, if you have 20 nodes in your tree and the pruning factor is 0.2, you will prune 0.2 * 20 = 4 nodes randomly from your tree. After pruning the tree, you will check its accuracy on the validation dataset. If the validation accuracy goes up, it implies the pruning improved your model. Otherwise, you might need to pick a different set of nodes for pruning.

- you will re-compute the training and test accuracy and output the summary on the screen as before. You should also output the plot of the post-pruned decision tree model.

  Post-Pruned Accuracy

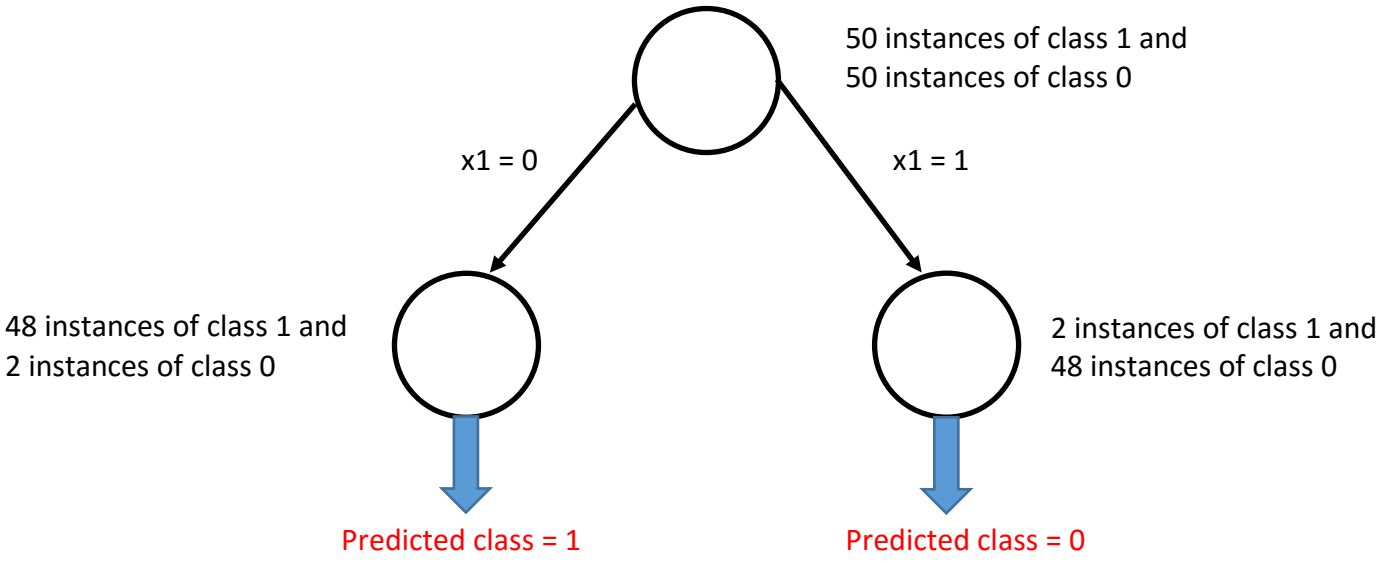
  Number of training instances = 100
  Number of training attributes = 5
  Total number of nodes in the tree = 16
  Number of leaf nodes in the tree = 6
  Accuracy of the model on the training dataset = 90.2%

  Number of validation instances = 50
  Number of validation attributes = 5
  Accuracy of the model on the validation dataset after pruning = 80.9%

  Number of testing instances = 20
  Number of training attributes = 5
  Accuracy of the model on the training dataset = 72.4%

Some important things and assumptions:

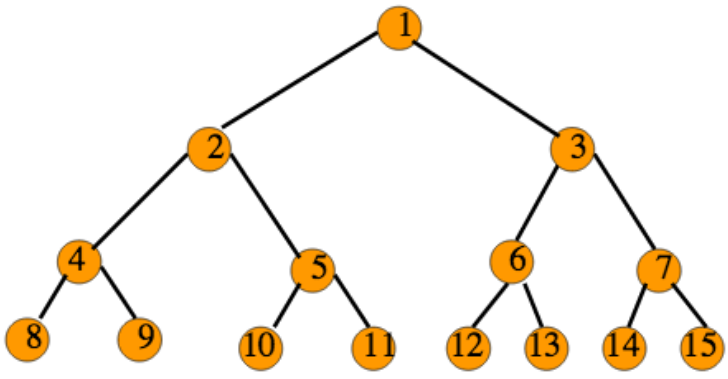
- Remember that entropy uses logarithm of base 2. You should implement $\log_2$ and not any other base. You can assume that $\log_2 0 = 0$.
- After constructing a tree, if you realize that there is a leaf node that contains data from more than 1 class i.e. it is not a pure node, then you should choose the most frequent class in that node and output that as the predicted class.



- If you reach a leaf node that contains no data instances, you are free to label it anyway you like (Toss a coin!).
- You can make any other reasonable assumptions, but mention them clearly in your report.
- Accuracy is defined as:

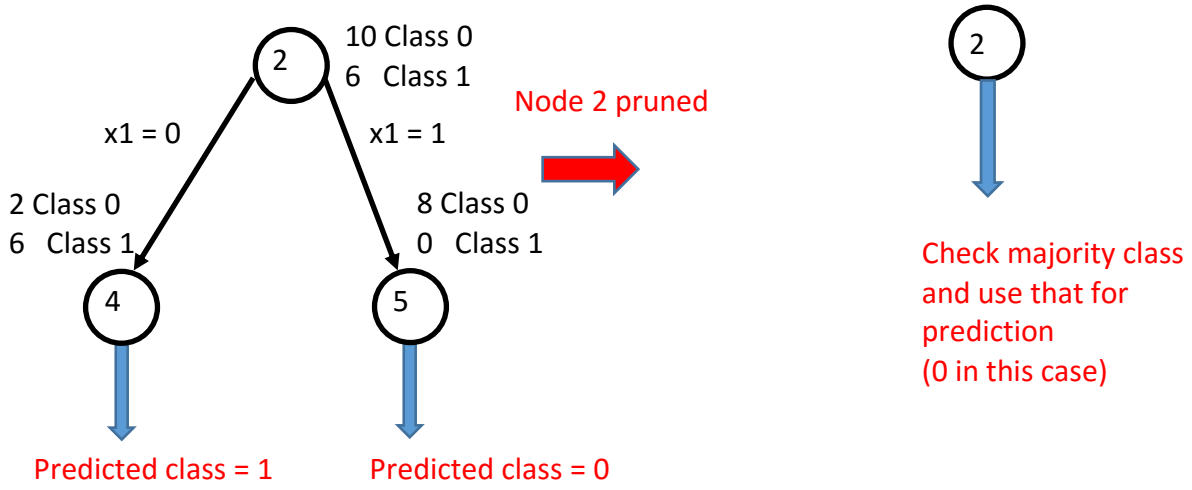
$$\text{Accuracy} = \frac{\text{Number of instances correctly classified}}{\text{Total number of instances}}$$

- For pruning, it would be helpful if you label the nodes. An example is shown below:



The above tree has 15 nodes. Suppose the pruning factor is 0.2, then you would randomly prune 3 nodes. Your program can simply choose 3 random nodes and remove them. The details of the pruning algorithm can be understood by reading the section 3.7.1.1 (Reduced Error Pruning) in the textbook:

For example, suppose you decide to prune node 5 below, you will simply delete the sub-tree rooted at it.



What to submit

- For part I, submit either a typed solution or a scanned, legible handwritten solution. If the TA cannot read your answers, you will not get any credit.
- For part II, submit the following:
  - README file indicating which language you used, how to compile and run your code.
  - source code (no executables)
  - A brief report indicating any assumptions that you made, your best results, what you accomplished, and what you learned. The report should clearly indicate the names of the team members.
  - Screenshots of 1 run of your program.
- Please create two folders for the two parts – parti and partii. Zip these two folders and submit on eLearning.
- Only 1 submission per team. If there is more than 1 submission, we reserve the right to grade any one and assign points on that basis.
- Submission on eLearning by the deadline. The link will go offline 2 days after the deadline.