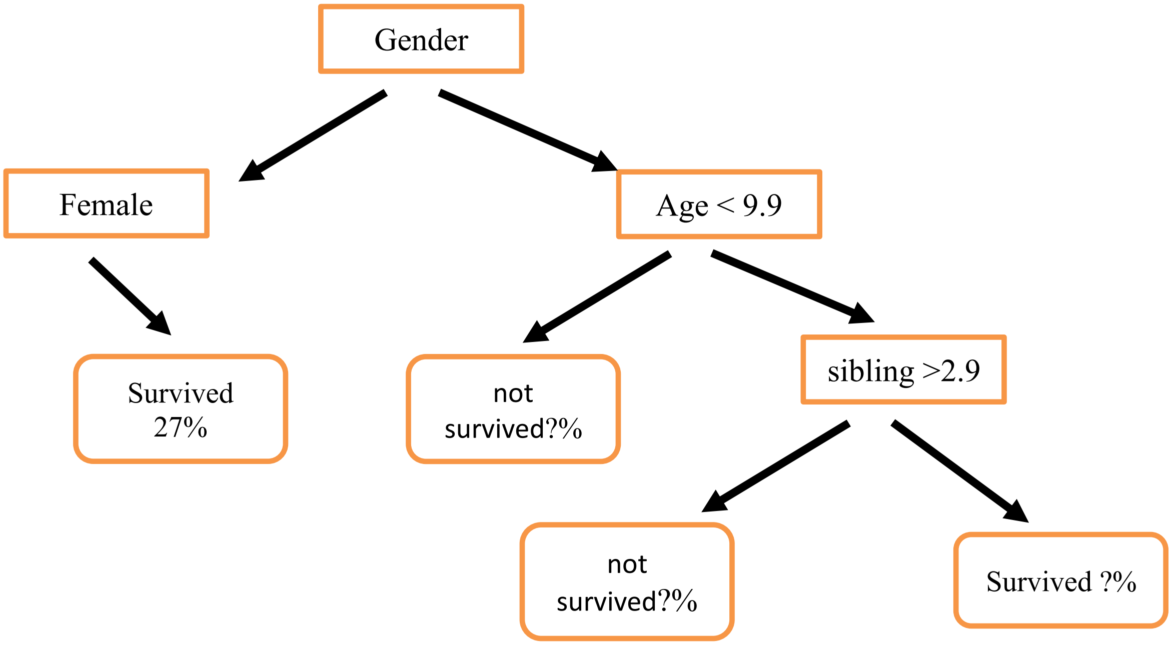
CP600 -Fall 2020

CP600 : Practical Algorithm Design
Assignment 2 -Programming
Due by Tuesday Nov 7, 2020 at 23:55

In this assignment, we will explore decision trees. A binary decision tree is a full binary tree to identify the class to which observed data belongs. The following example illustrates this concept.



This example shows a decision tree to predict whether a given person survived at Titanic or not by using their gender, age and number of siblings they had. They have been obtained based on a large number of samples (for which age, gender and siblings were known).

We try to classify the titanic passengers based on these features to the status of survived/ not survived. We do so by applying the rules of the decision tree on a large number of observations. Also, it is possible to count how many samples of each class reach a given leaf. For example, for the leaf corresponding to the decision [Female, Age <9, sibling <2.9], it was observed that in 73% of cases, the sample reaching this leaf was a person 'Not survived' and in 27% of the cases, it was a person survived. So, if an instance that we want to classify enters the root and ends on this leaf, the class assigned will be 'Not survived', since there were a higher percentage of samples (73%) belonging to this class that arrived at this leaf.

To perform the classification, the decision tree uses input feature vectors that have a certain size (3 in our example, age and gender and number of siblings). The decision is to determine the membership class; in our example, there are 2 classes (Survived or not survived), but in general there may be more.

The decision tree consists of nodes that use one of the entries of the decision vector and applies a threshold. Depending on the result of the comparison, we branch left or right. A feature vector enters the tree by the root. It follows a series of comparisons leading to a leaf of the tree.

Instructions:

The file `BaseDecisionTree.java` provides an implementation for a decision node. Each node specifies an index (`featureIndex`) in the feature vector and a threshold (`threshold`) to be used in the comparison. The leaves of the tree are dummy nodes without comparison, which return the class with the highest probability for that node. These probabilities are obtained either by specifying them directly or by accumulating the results obtained by using pre-classified samples (`classFreq` and `total`).

The `BaseDecisionTreeTester.java` builds the decision tree is shown in the above figure and tests its results with two sets of titanic data:

- A set of simulated data for probabilities are used to:
  - build the tree, probabilities for each class output (in percentage) for each external(leaf) node are given.
  - test this tree with 5 samples.
- Real data in which you use to:
  - complete the unknown probabilities in the above tree to build your own tree, the first probability is given to you in the tree (27% for female survivals).
  - Test your work. A set of tests is given and commented. You can comment them out once you finish your code to test your tree.

To complete the unknow probabilities in the figure and test your tree, you are given the following files in the zip folder:

- The source code `BaseDecisionTree.java` in which you should fill the incomplete methods with your own code.
- A file (`input.txt`) containing the real titanic input to calculate the probabilities for your decision tree.
- A file (`output.txt`), which you should use to calculate the accuracy of your tree.

Please use the `BaseDecisionTree.java` and 3 text files to complete `BaseDecisionTree` class by adding the requested operations:

1. To build your tree, you are required to use the input data file `input.txt` to calculate the probabilities for your decision tree for the real titanic data. The file has four columns: 3 for input features (gender, age and #siblings) while the last column represents the correct answer survived/not survived,
2. To test your tree, calculate the accuracy for your decision tree by using use the files, `output.txt` (the file has three columns: gender, age and #siblings) that has samples of the titanic data and `output_answers.txt` (has one column, the output class) which has the correct answers for each sample.

You must submit a short report (suggested max 2 pages) in file report.pdf that contains:

- A description of how you obtain the solution.
- Optional: any additional information to be seen by the marker.

We ask you to submit a zip file named studentNumber_a2.zip containing:

- your code: all commented .java files required to run your program. Make sure to write your name and student number at the beginning of each file.
- a file report.pdf containing the report