You must implement a Matlab function or a Python executable file called decision\_tree. Your function should be invoked as follows:

decision\_tree(<training\_file>, <test\_file>, <option>, <pruning\_thr>)

If you use Python, just convert the Matlab function arguments shown above to command-line arguments. The arguments provide to the function the following information:

The arguments provide the following information:

* The first argument, <training\_file>, is the path name of the training file, where the training data is stored. The path name can specify any file stored on the local computer.
* The second argument, <test\_file>, is the path name of the test file, where the test data is stored. The path name can specify any file stored on the local computer.
* The third argument, <option>, can have four possible values: optimized, randomized, forest3, or forest15. It specifies how to train (learn) the decision tree, and will be discussed later.
* The fourth argument,<pruning\_thr>, is a number greater than or equal to 0, that specifies the pruning threshold.

The training and test files will follow the same format as the text files in the [UCI datasets](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/uci_datasets) directory. A description of the datasets and the file format can be found [on this link](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/uci_datasets/dataset_description.html). For each dataset, a training file and a test file are provided. The name of each file indicates what dataset the file belongs to, and whether the file contains training or test data. Your code should also work with ANY OTHER training and test files using the same format as the files in the [UCI datasets](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/uci_datasets) directory.

As the [description](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/uci_datasets/dataset_description.html) states, **do NOT use data from the last column (i.e., the class labels) as features**. In these files, all columns except for the last one contain example inputs. The last column contains the class label.

**Training Stage**

The first thing that your program should do is train a decision tree or a decision forest using the training data. What you train and how you do the training depends on the came of the third argument, that we called <option>. This option can take four possible values, as follows:

* optimized: in this case, at each non-leaf node of the tree (starting at the root) you should identify the optimal combination of attribute (feature) and threshold, i.e., the combination that leads to the highest information gain for that node.
* randomized: in this case, at each non-leaf node of the tree (starting at the root) you should choose the attribute (feature) randomly. The threshold should still be optimized, i.e., it should be chosen so as to maximize the information gain for that node and for that randomly chosen attribute.
* forest3: in this case, your program trains a random forest containing three trees. Each of those trees should be trained as discussed under the "randomized" option.
* forest15: in this case, your program trains a random forest containing 15 trees. Each of those trees should be trained as discussed under the "randomized" option.

**Training Phase Output**

After you learn your tree or forest, you should print it. Every node must be printed, in breath-first order, with left children before right children. For each node you should print a line containing the following info:

* tree ID. If you are learning a single tree, the ID is 1. If you are learning multiple trees, their ID range from 1 to the number of trees, and should be printed in increasing order of their ID.
* node ID. The root has ID 1. If a node has ID = N, then its left child has ID 2\*N and its right child has ID 2\*N + 1.
* a feature ID, specifying which attribute is used for the test at that node. This is a number starting from 1, indicating the position of the column that contains values for that attribute. If the node is a leaf node, print -1 for the feature ID.
* a threshold that, combined with the feature ID specifies the test for that node. If feature ID = X and threshold = T, then objects whose X-th feature has a value LESS THAN T are directed to the left child of that node. If the node is a leaf node, print -1 for the threshold.
* information gain achieved by the test chosen for this node. For a leaf node, you should consider the information gain at that node to be 0.

To produce this output in a uniform manner, use this printing statement:

fprintf('tree=%2d, node=%3d, feature=%2d, thr=%6.2f, gain=%f\n', tree\_id, node\_id, feature\_id, threshold, gain);

**Classification Stage**

For each test object you should print a line containing the following info:

* object ID. This is the line number where that object occurs in the test file. Start with 1 in numbering the objects, not with 0.
* predicted class (the result of the classification). If your classification result is a tie among two or more classes, choose one of them randomly.
* true class (from the last column of the test file).
* accuracy. This is defined as follows:
  + If there were no ties in your classification result, and the predicted class is correct, the accuracy is 1.
  + If there were no ties in your classification result, and the predicted class is incorrect, the accuracy is 0.
  + If there were ties in your classification result, and the correct class was one of the classes that tied for best, the accuracy is 1 divided by the number of classes that tied for best.
  + If there were ties in your classification result, and the correct class was NOT one of the classes that tied for best, the accuracy is 0.

To produce this output in a uniform manner, use this printing statement:

fprintf('ID=%5d, predicted=%3d, true=%3d, accuracy=%4.2f\n', object\_id, predicted\_class, true\_class, accuracy);

After you have printed the results for all test objects, you should print the overall classification accuracy, which is defined as the average of the classification accuracies you printed out for each test object. To print the classification accuracy in a uniform manner, use this printing statement:

fprintf('classification accuracy=%6.4f\n', classification\_accuracy);