CSE5243 **Student Name:** _____
Spring 2025 **Student ID:** _____
Programming Assignment 1
Due date: Tuesday, Feb 18, 2:19pm

Instructions:

In this programming assignment, you will implement a decision tree. The total point is 115 and you can get another 6 points as bonus. Good luck!

Note: This is not an easy programming assignment. Start earlier!

Grade Table (for TA/grader use only)

Question	Points	Score
1	5	
2	5	
3	70	
4	25	
5	10	
Total:	115	

You will be given three input files and you will program and test a decision tree using the data in the input files. The first input file train_data.txt has n lines, and in each line there are m numbers (either 0 or 1), delimited by an empty space. Each of the n lines has a data point of m features, and each of the m numbers in a line is the value of that data point on the corresponding feature. The second input file train_label.txt has n lines, and each line has exactly one number representing the class label of the corresponding data point in train_data.txt. For example, if the i-th line in train_label.txt has 2, it means the i-th data point in train_data.txt has a class label 2. The data points from train_data.txt and their labels from train_label.txt represent the training data. You will train a decision tree classifier from the training data. You will test new testing data using the decision tree classifier. The testing data are given in the input files test_data.txt, which has the same format as in train_data.txt.

1. (5 points) You need to implement a simple user interface (via command line) such that users can input training/testing file names and other parameters. For example, if your executable is named trainDT, then the command line should look like

   trainDT.py -train_data train_data.txt -train_label train_label.txt -test_data test_data.txt -nlevels nl -pthrd p -impurity xxx -pred_file output.txt

   where "-nlevels" is a parameter which constrains the levels of the decision tree, and a positive number "nl" should be given for this option; "-pthrd" is a parameter which constrains the maximum impurity for a node and a positive number p should be given for this option; "-impurity" is the method used to calculate impurity, which takes "xxx" as either "gini" or "entropy"; "-pred_file" is the prediction file name, which takes "output.txt" as the file name.

   This URL `https://docs.python.org/2/library/argparse.html` may give you some idea on how to implement the options via command line. You just need

   - import argparse
   - parser = argparse.ArgumentParser()
   - parser.add_argument('-train_data')
   - args = parser.parse_args()
   - train_file = str(args.train_data)

   to get the input file name, etc.

2. (5 points) You need to read in the training data and labels, and testing data from the input files. You need to store the data into proper data structures. This URL `https://www.numpy.org/devdocs/reference/generated/numpy.genfromtxt.html` may be helpful. You just need

   - import numpy as np
   - X = np.genfromtxt(file_name, delimiter=' ')

to get the data from file_name into a matrix X, where numpy is a Python library that supports for large, multi-dimensional arrays and matrices and their operations.

3. (70 points) You need to build a decision tree from the training data. The decision tree you are going to build is a binary tree, that is, each node will have at most two children. Each internal node has a decision rule (e.g., attribute == 0?), and its left child has all the data points of its parent on which the decision is true (e.g., attribute == 0), and its right child has all the data points of its parent on which the decision is false (e.g., attribute == 1). All the leaf nodes should have the data points which are routed from the root according to the decision rules along the path to the leaf nodes.

   - (10 points) You need to properly design a tree node data structure, which has indices to the data points (node.data_idx) that reside at the node (4 points), impurity method used to split nodes (node.impurity_method, 1 point), a decision rule (node.dfearure, 1 point), the impurity value of the node (node.impurity, 1 point), the level of the node (node.nlevels, 1 point), the number of features (node.mfeatures, 1 point) and the majority class label for all the resident data points (node.class, 1 point). The decision rule can be simply represented by a feature index. For example, if the decision feature index is 2 at a node, it means the decision rule is "feature 2 == 0?" by default. This URL `https://www.tutorialspoint.com/python/python_binary_tree.htm` may give you some idea on how to design a simple node in a binary tree.
   - (10 points) You need to implement node.buildDT() to build a decision true, starting from a root. The following pseudo code may give you some idea on how to build a tree recursively.

```
1
2     def buildDT(data, label, impurity_method, nl, p):
3
4           # 0 is the level of the node
5           DT = initNode(data_idx, impurity_method, 0)
6
7           DT.splitNode(nl, p)
8
9     return DT
```

   - (15 points) That is, you need to have a constructor node.initNode() which builds a root node from "data_idx" (i.e., indices to the training data), calculates the node impurity using the "impurity_method" (will be discussed later), and properly initialize other member variables in the node.
   - (20 points) You also need to implement a split function node.splitNode() which splits the node based on impurities. The following pseudo code may give you some idea on how to split a node.

```
1
2     def splitNode(nl, p):
3
4
```

```
        if self.nlevels < nl and self.impurity > p:

            maxGain = -1
            splitFeature = -1

            for feature_i in range(0, self.nfeatures-)

                Pleft  = calculateIP(data with feature_i == 0)
                Pright = calculateIP(data with feature_i == 1)

                # you need to implement how to calculate impurity
                # after split in the following line (5 points)
                # Pleft is the impurity from left child,
                # Pright is the impurity from right child
                M = impurity after split
                Gain = self.impurity - M

                if Gain > maxGain:
                    maxGain = Gain
                    splitFeature = feature_i

            self.dfeature = splitFeature

            # you need to figure out which data should
            # be in the left child
            # and which in the right child
            # if split using splitFeature (5 points)
            data_idx_left = data on the left
            data_idx_right = data on the right

            self.left_child =
            initNode(data_idx_left, impurity_method, self.nlevels+1)
            self.right_child =
            initNode(data_idx_right, impurity_method, self.nlevels+1)

            self.left_child.splitNode(nl, p)
            self.right_child.splitNode(nl, p)

    return
```

- (5 points) You need to implement calculateIP() as in the following pseudo code:

```
    def calculateIP(data_idx):

    if impurity_method is ''gini'':
        P = calculateGINI(data_idx)

    else if impurity_method is ''entropy'':
        P = calculateEntropy(data_idx)

    return P
```

- (10 points) That is, in calculateIP(), you need to implement calculateGINI() (5

points) and calculateEntropy() (5 points), which calculate the gini index and entropy of current node.

Note that Node(), buildDT(), splitNode() and calculateIP() should be member functions of the tree node class.

4. (25 points) You need to implement DT.classify() which predicts the each of labels for testing data and output the labels into an output file. Each line of the output file should have exactly one number representing the predicted label of the corresponding testing data point.

```
1
2     def classify(test_data, output_file)
3
4     foreach data in test_data
5     # classify data using the decision tree (20 points)
6     # and output the predicted labels into output_file (5 points)
```

This URL `https://www.tutorialspoint.com/python/python_files_io.htm` may give you some idea on how to do file I/O in python.

5. (10 points) You need to implement a main() function

Bonus question (6 points) If you could print out precision, recall and accuracy of the prediction from the decision tree for class 1. That is, all other classes are considered as not class 1, and thus you will have a binary classification problem. A test_label.txt file is provided for this purpose, which has the true labels of the testing data.

What to submit:

1. Your source code in a zipped file. You should name your zipped file as X_Y_PA1.tar.gz where X is your first name (capital letters) and Y is your last name (capital letters). Please following the naming schemes specified, otherwise, **20** points will be taken off. This is for the TA/graders to quickly find your results.
2. It is mandatory that at least 1/4 of the lines (not including empty lines) in your code should be dedicated to comments. Otherwise, **20%** of the points that you will get will be taken off (non-genotiable!)