CSC 3520
Machine Learning
Florida Southern College

HW2: Decision Trees
Due: Wednesday, February 21, 2024

The purpose of this assignment is to provide an opportunity for you to:
- train and test a decision tree model to make predictions on an important healthcare issue
- increase your Python programming skills related to machine learning and associated libraries
- demonstrate the ability to critically analyze performance results for a real-world problem

Parkinson's disease (PD) is a common progressive disorder of the central nervous system that affects millions of people around the world. The exact cause of PD is unknown and there is no known cure at this time. Treatment is usually focused on mitigating symptoms that include tremors, movement loss, and speech impairment. Recently, researchers have discovered that certain measurements of dysphonia (difficulty in speaking) can help discriminate between healthy individuals and those suffering from PD. For this assignment, you will work in pairs to develop Python code that uses decision trees to predict whether someone has Parkinson's disease based on dysphonia measurements.

The data for this problem has been provided for you. There are 185 samples for training and 10 samples for testing. Each sample contains 22 attributes related to dysphonia; for details about these measurements, see https://archive.ics.uci.edu/ml/datasets/Parkinsons or [1]. The relevant data files are listed below.

`attributes.txt`	List of dysphonia measurement variables used for classification.
`training_data.txt` `testing_data.txt`	Dysphonia measurement values for each training/testing sample. Each row is a unique sample containing 22 comma-separated floating-point values.
`training_labels.txt` `testing_labels.txt`	Each line corresponds to a sample from the training/testing set and determines whether that individual is healthy (0) or has Parkinson's (1).

1. Write a Python program to train and test a decision tree using information gain on the data described above. Starter code has been provided for you (`decisiontree.py`). Sections that require editing are either marked by the text `***MODIFY CODE HERE***` or a variable set to `-1`. A summary of the required modifications is given below, listed in the order each task appears in the starter code:

   - Change the `ROOT` variable to point to the data directory on your local computer
   - Load the training/testing data/labels and attributes from the appropriate files into the variables `xtrain`, `ytrain`, `xtest`, `ytest` and `attributes`
   - Create a `DecisionTreeClassifier` object using appropriate parameters and fit the tree to the training data
   - Show the tree using the `plot_tree` function. In your visualization, at a minimum, you must include feature names (i.e. `attributes`), class names (i.e. `parkinsons` vs `healthy`), filled nodes, and rounded nodes. Note that the code is already setup to "save" and "show" this figure using the `ArgumentParser` options provided.
   - Uncomment the two lines of code that automatically extract the index of the root node and corresponding threshold from the trained classifier. You can also delete the placeholder code that

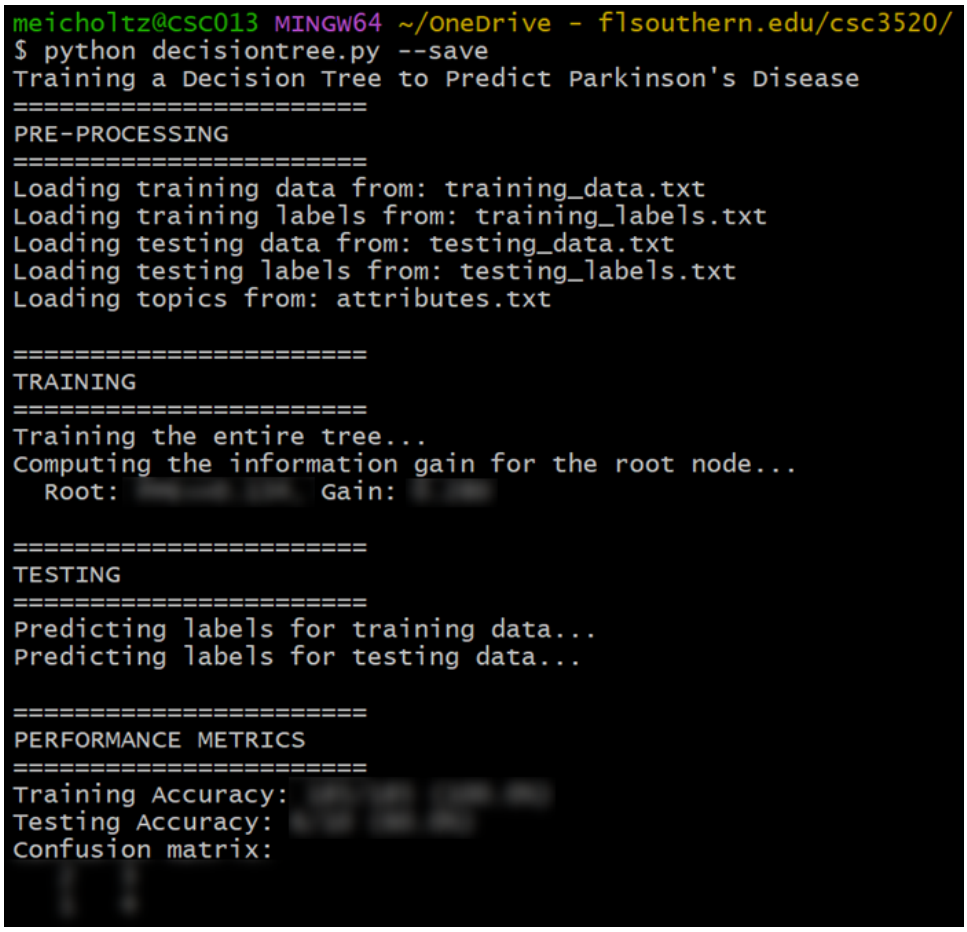
sets these variables to `-1`.

- Use the trained classifier to predict labels for both the training and testing data
- Compute and display the training and testing accuracy
- Compute and display the confusion matrix on the testing data
- Complete the `information_gain` function that takes training data (`x`) and labels (`y`) along with an attribute `index` and threshold (`thold`) and returns the gain of the label given the binary feature of the training attribute `<=` the threshold. The purpose of this function is to demonstrate an understanding of the fundamental principle underlying the training process of the `DecisionTreeClassifier`. You are encouraged to create additional functions as needed to complete the task (e.g. `entropy`?). You are also encouraged to verify your result by manually computing the gain using the saved tree image that is generated by your code.

Here are some functions from the imported libraries that *may* be useful:

`matplotlib.pyplot`	`close, savefig, show, subplots`
`numpy`	`array, loadtxt, log2, mean, sum`
`sklearn.tree`	`DecisionTreeClassifier, plot_tree`
`sklearn.metrics`	`confusion_matrix`

For reference, the output of a successful program is pictured below.



2. After completing the program outlined above, discuss the results in a separate written document. Some questions to consider include (not an exhaustive list):

   - What is the depth of the tree? Consider the depth to be the maximum number of *decision nodes* (do not count leaf nodes) that could be used to classify a single example
   - How many leaf nodes are there?
   - How many of the 22 attributes are used in the tree? What does this mean about the other attributes?
   - Consider the confusion matrix for this tree on the test data. Does the tree more often misclassify people who are healthy or people who have Parkinson's? Practically speaking, which bias would you rather have (misclassifying healthy or sick people)?
   - Does your decision tree overfit your data? How do you know?
   - How could the classifier be improved to increase the overall performance on test data? Feel free to experiment with design parameters.
   - ...

   At a minimum, your analysis must be at least 2 paragraphs in length.

Assessment

This assignment is graded as complete/incomplete. In order to achieve completion, your submission must include the following deliverables:

- The completed `decisiontree.py` code
- A PDF containing your analysis of the results
- A PNG file of the trained decision tree; this should be automatically generated by your program when using the `--save` flag at the command line

In addition, your submission will be evaluated using several factors, including, but not limited to:

- **Correctness:** Does your program run without error and does it actually implement the proposed machine learning algorithm? Test your code thoroughly!
- **Content:** Does your code adequately solve each of the tasks outlined above?
- **Clarity:** Is your code easy to follow and understand?
- **Performance:** How well does your machine learning classifier perform? Are the results reasonable?
- **Analysis:** Did you properly interpret the results using critical thinking and thoughtful discussion?

References

[1] Little, M.A., McSharry, P.E., Hunter, E.J., Spielman, J. and Ramig, L.O., 2009. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Transactions on Biomedical Engineering*, 56(4), pp.1015-1022.