Assignment #2: Decision Trees and Random Forests
CSCI 373 Spring 2024 Oberlin College
Due: Friday March 15 at 11:59 PM

Background

Our second assignment this semester has three main goals:

1. Practice using popular Python libraries for creating machine learning models for classification problems,
2. Conduct extensive experiments evaluating machine learning performance, and
3. Practice inspecting the information learned by a machine learning algorithm.

Gitting Started

You can get started on the assignment by following this link:

https://classroom.github.com/a/42rKmsKG

Data Sets

For this assignment, we will learn from five pre-defined data sets:

1. **banknotes.csv**: A data set describing observed measurements about banknotes (i.e., cash) under an industrial print inspection camera. The task in this data set is to predict whether a given bank note is authentic or a forgery. The four attributes are each continuous measurements. This data set comes the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/banknote+authentication
2. **mnist1000.csv**: A data set of optical character recognition of numeric digits from images. Each instance represents a different grayscale 28x28 pixel image of a handwritten numeric digit (from 0 through 9). The attributes are the intensity values of the 784 pixels. Each attribute is ordinal (treat them as continuous for the purpose of this assignment) and a nominal label. This version of MNIST contains 1,000 instances of each handwritten numeric digit, randomly sampled from the original training data for MNIST. The overall MNIST data set is one of the main benchmarks in machine learning: http://yann.lecun.com/exdb/mnist/. It was converted to CSV file using the python code provided at: https://quickgrid.blogspot.com/2017/05/Converting-MNIST-Handwritten-Digits-Dataset-into-CSV-with-Sorting-and-Extracting-Labels-and-Features-into-Different-CSV-using-Python.html
3. **occupancy.csv**: A data set of measurements describing a room in a building for a Smart Home application. The task in this data set is to predict whether or not the room is occupied by people. Each of the five attributes are continuous measurements. This data set comes the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+

4. **penguins.csv**: A data set describing observed measurements of different animals belonging to three species of penguins. The four attributes are each continuous measurements, and the label is the species of penguin. Special thanks and credit to Professor Allison Horst at the University of California Santa Barbara for making this data set public: see this Twitter post and thread with more information (https://twitter.com/allison_horst/status/1270046399418138625) and GitHub repository (https://github.com/allisonhorst/palmerpenguins).
5. **seismic.csv**: A data set of measurements describing seismic activity in the earth, measured from a wall in a Polish coal mine. The task in this data set is to predict whether there will be a high energy seismic event within the next 8 hours. The 18 attributes have a mix of types of values: 4 are ordinal attributes, and the other 14 are continuous. The label is "no event" if there was no high energy seismic event in the next 8 hours, and "event" if there was such an event. This data set comes the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/seismic-bumps

The file format for each of these data sets is the same as in Homework #1:

- The first row contains a comma-separated list of the names of the label and attributes
- Each successive row represents a single instance
- The first entry (before the first comma) of each instance is the correct label we want to the computer to learn to predict, and all other entries (following the commas) are attribute values.
- Some attributes are integers and others are real numbers. Each label is a string.

Program

Your assignment is to write a program called **trees.py** that behaves as follows:

1) It should take four parameters as input from the command line:

    a. The path to a file containing a data set
    b. The number of trees to use (either 1 for a single Decision Tree or more than 1 to use a Random Forest)
    c. The percentage of instances to use for a training set
    d. An integer to use as a random seed

    For example, I might run

    ```
    Python trees.py mnist1000.csv 100 0.75 12345
    ```

    which will train a Random Forest with 100 trees to learn how to classify the mnist1000.csv data set using a random seed of 12345, where 75% of the data will be used for training (and the remaining 25% will be used for testing)

2) Next, the program should read in the data set as a set of instances, which should be split into training and test sets (using the random seed input to the program), similar to the process in Homework 1
3) If the user input 1 for the number of trees (the second command line argument), then a single Decision Tree classifier should be trained using the training set created in Step 2. If instead, the user input a number greater than 1 for the number of trees, then a Random Forest classifier with the input number of tree should be trained on the training set.
4) Next, predictions should be made for each instance in the test set created in Step 2, using the model trained in Step 3.
5) A confusion matrix should be created based on the predictions made during Step 4, then the confusion matrix should be output as a file with its name following the pattern: `results_<DataSet>_<NumTrees>t_<TrainingPercentage>p_<Seed>.csv` (e.g., `results_mnist1000_100t_75p_12345.csv`)

Please note that you **are** allowed to reuse your code from Homework 1 and any labs for generating random test/training sets, as well as for creating output files.

Program Output

The file format for your output file should be the same as in Homework 1. Please refer back to that assignment for more details.

As in Homework 1, the output for your program should be consistent with the random seed. That is, if the same seed is input twice, your program should output the exact same confusion matrix.

Programming Languages

In this class, we are using the **Python** programming language for all of our labs and assignments. For this assignment, you are allowed to use any Python libraries (e.g., `pandas`, `numpy`, `scikit-learn`) and not only those originally built into Python. In particular, you should use `scikit-learn` to train DecisionTreeClassifer and RandomForestClassifier.

Research Questions and Experience

Please use your program to answer the eight research questions in the provided `README.md` file. There are also three questions at the end of the `README.md` file that you should answer about your experience completing the assignment.

Please remember to commit your solution code, image files, and `README.md` file to your repository on GitHub. You do not need to wait until you are done with the assignment to commit your code. ***Make sure to document your code***, explaining how you implemented the different components of the assignment.

Honor Code

Different from the first homework assignment, **each student is allowed to work with *one partner* to complete this assignment**. Groups are also encouraged to collaborate with one another to discuss the abstract design and processes of their implementations. However, sharing code or answers to the research questions (either electronically or looking at each other's files) between groups is not permitted.

Grading Rubric

Your solution and `README.md` will be graded based on the following rubric:

Followed input and output directions: /5 points
Properly read in and split the data into training and test sets: /5 points
Correctly trains a single Decision Tree: /10 points
Correctly trains a Random Forest of multiple trees: /10 points
Correctly makes predictions and creates a confusion matrix: /10 points
Correctly answered the research questions: /50 points
Provided requested README information: /5 points
Appropriate code documentation: /5 points

By appropriate code documentation, I mean including a header comment at the top of each file explaining what the file provides, as well as at least one comment at the top of each function explaining the purpose of the function (inline comments are also welcome).