Assignment #1: k-Nearest Neighbors
CSCI 373 Spring 2024 Oberlin College
Due: Wednesday February 28 at 11:59 PM

Background

Our first assignment this semester has three main goals:

1. Implement and evaluate our first machine learning algorithm,
2. Conduct our first experiments to evaluate machine learning performance, and
3. Practice using GitHub for source code management

Gitting Started

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

https://classroom.github.com/a/Ep4kCi5E

Data Sets

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

1. **monks1.csv**: A data set describing two classes of robots using all nominal attributes and a binary label. This data set has a simple rule set for determining the label: `if head_shape = body_shape V jacket_color = red, then yes, else no.` Each of the attributes in the monks1 data set are nominal. Monks1 was one of the first machine learning challenge problems (http://www.mli.gmu.edu/papers/91-95/91-28.pdf). This data set comes from the UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/MONK%27s+Problems
2. **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).
3. **mnist100.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 100 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

4. **mnist1000.csv**: The same as mnist100, except containing 1000 instances of each handwritten numeric digit.

The file format for each of these data sets is as follows:

- 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 teach the computer to predict, and all other entries (following the commas) are attribute values.
- Some attributes are strings (representing nominal values), some are integers, and others are real numbers. Each label is a string.

Program

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

1) It should take five parameters as input from the command line:
   a. The path to a file containing a data set
   b. The name of the distance function to use: either H for Hamming distance or E for Euclidian distance
   c. The value of *k* to use in the *k*-Nearest Neighbors algorithm
   d. The percentage of instances to use for a training set
   e. An integer to use as a random seed

   For example, I might run

   ```
   python knn.py mnist100.csv E 1 0.75 12345
   ```

   which will perform 1-Nearest Neighbors on the mnist100.csv data set using the Euclidian distance function and 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) To begin execution, the program should read in the data set as a collection of instances
3) Then, the instances should be randomly split into training and test sets (using the percentage and random seed input to the program)
4) Next, predictions should be made for each instance in the test set created in Step 3, using the training set as the instances to compare in the *k*-Nearest Neighbors algorithm.
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>_<k>_<Seed>.csv` (e.g., `results_mnist100_1_12345.csv`)

Program Output

The file format for your output file should be as follows:

- The first row should be a comma-separated list of the possible labels in the data set, representing the list of possible predictions your program could make. **This row should end in a comma** (before the newline \n character).
- The second row should be a comma-separated list of the counts of many times each of the different labels were predicted for instances in the test set whose **true label** is the first possible label, ending with the name of the first possible label (and not a final comma).
- The third row should be a comma-separated list of the counts of many times each of the different labels were predicted for instances in the test set whose **true label** is the second possible label, ending with the name of the second possible label (and not a final comma).
- Etc. for the remaining possible true labels

For example, the confusion matrix:

Predicted Label			
Yes	No		
200	100	Yes	Actual
50	250	No	Label

would be output as:

```
Yes,No,
200,100,Yes
50,250,No
```

The output for your program should be consistent with the random seed. That is, if your run your program twice with the same command line arguments, then 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 first assignment, you are allowed to use any library built-in to Python (i.e., that does not need to be installed using `pip`). Allowed libraries include `random`, `sys`, `math`, and `csv`. You should *not* use any external libraries (e.g., `pandas`, `numpy`, `scikit-learn`); we will practice working with external libraries in lab and then use them for our later assignments. If in doubt, please feel free to ask Adam if a library is allowed on this assignment.

Research Questions and Experience

Please use your program to answer the four research questions in the provided `README.md` file. There is also an optional bonus question (worth an additional 5 points) that you can choose to answer, if you want to conduct an additional experiment and evaluate the importance of different features in the penguins.csv data set.

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, results 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; it is good practice to commit and push not only after each coding session, but maybe after hitting important milestones or solving bugs during a coding session. ***Make sure to document your code***, explaining how you implemented the different components of the assignment.

Honor Code

Each student is to complete this assignment individually. However, students are encouraged to collaborate with one another to discuss the abstract design and processes of their implementations. For example, please feel free to discuss the pseudocode for the learning algorithm. You might also want to discuss the processes used to generate the training and test sets from the read in data set. Or, you might need to discuss how to work with the input and output files. At the same, since this is an individual assignment, no code can be shared between students, nor can students look at each other's code.

Grading Rubric

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

Followed input directions: /5 points
Properly created training and test sets: /15 points
Correctly implemented the k-Nearest Neighbors algorithm: /25 points
Runtime efficiency: /10 points
Followed output directions: /10 points
Correctly answered research questions: /25 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).