Programming Assignment 8

Summary

In this assignment you will implement k-nearest neighbor classifiers.

NOTE: This is an **optional** assignment, This will only be added to your programming assignment average if it Improves it. If you do not make a submission or if your points awarded does not increase your average, this assignment will be ignored.

Command-line Arguments

You must implement a program that learns a naive Bayes classifier for a classification problem, given some training data and some additional options. In particular, your program can be invoked as follows:

```
knnclassify <training_file> <test_file> <k>
```

Both the training file and the test file are text files, containing data in tabular format. Each value is a number, and values are separated by white space. The i-th row and j-th column contain the value for the j-th feature of the i-th object. The only exception is the LAST column, that stores the class label for each object.**Make sure you do not use data from the last column (i.e., the class labels) as attributes (features) in your decision tree.**

Example files that can be passed as command-line arguments are in the datasets directory. That directory contains three datasets, copied from the UCI repository of machine learning datasets:

- The `pendigits` dataset, containing data for pen-based recognition of handwritten digits.
  - 7494 training objects.
  - 3498 test objets.
  - 16 attributes.
  - 10 classes.
- The `satelite` dataset. The full name of this dataset is Statlog (Landsat Satellite) Data Set, and it contains data for classification of pixels in satellite images.
  - 4435 training objects.
  - 2000 test objets.
  - 36 attributes.
  - 6 classes.
- The `yeast` dataset, containing some biological data whose purpose I do not understand myself.
  - 1000 training objects.
  - 484 test objets.
  - 8 attributes.
  - 10 classes.

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.

Note that, for the purposes of your assignment, it does not matter at all where the data came from. One of the attractive properties of decision trees (and many other machine learning methods) is that they can be applied in the exact same way to many different types of data, and produce useful results.

Training

For this phase you should classify the test data using a k-nearest neighbor classifier. The value of k is specified by the third command-line argument.

In your k-nearest neighbor classifier, you should use the following guidelines:

- Each attribute should be normalized, separately from all other attributes. Specifically, each attribute should be transformed

using function F(v) = (v - mean) / std, using the mean and std of the values of that attribute on the training data.
- Use the L_2 distance (the Euclidean distance) for computing the nearest neighbors.

There is no need to output anything for the training phase.

Classification

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 0 in numbering the objects, not with 1.
- 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).
- ID (index) of the nearest neighbor among training objects. Numbering of training objects should start at 0.
- Distance to the nearest neighbor among training objects. This should be the Euclidean distance, applied after normalizing the attributes.
- 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 these printing statements:

- For C or C++, use:

```
printf("ID=%5d, predicted=%3d, true=%3d, nn=%5d, distance=%7.2lf, accuracy=%4.2lf\n",
        object_id, predicted_class, true_class, nn_index, distance, accuracy);
```

- For Java, use:

```
System.out.printf("ID=%5d, predicted=%3d, true=%3d, nn=%5d, distance=%7.2f, accuracy=%4.2f\n",
                  object_id, predicted_class, true_class, nn_index, distance, accuracy);
```

- For Python or any other language, just make sure that you use formatting specifies that produce aligned output that matches the specs given for C and Java.

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 these printing statements:

- For C or C++, use:

```
printf("classification accuracy=%6.4lf\n", classification_accuracy);
```

- For Java, use:

```
System.out.printf("classification accuracy=%6.4f\n", classification_accuracy);
```

- For Python or any other language, just make sure that you use formatting specifies that produce aligned output that matches the specs given for C and Java.

Grading

- 30 points: Correct normalization of attributes.
- 60 points: Correct implementation of k-nearest neighbor classifiers.
- 10 points: Following the specifications in producing the required output.

How to submit

Submissions should be made using Blackboard. Implementations in Python, C, C++, and Java will be accepted. If you would like to use another language, please first check with the instructor via e-mail. Points will be taken off for failure to comply with this requirement.

Submit a ZIPPED directory called `programming-assignment8.zip` (no other forms of compression accepted, contact the instructor or TA if you do not know how to produce .zip files). The directory should contain:

- Source code for the programming part. Including binaries is optional.
- A file called readme.txt, which should specify precisely:
  - Name and UTA ID of the student.
  - What programming language is used.
  - How to run the code, including very specific compilation instructions. Instructions such as "compile using g++" are NOT considered specific. Providing all the command lines that are needed to complete the compilation on omega is specific.

Insufficient or unclear instructions will be penalized by up to 20 points. Code that does not run on omega machines gets AT MOST half credit, unless you obtained prior written permission.