CS 422 Homework Assignment 2

Due: 11:59pm, Thursday, March 1st

This assignment is scored out of 64. It consists of 5 questions. When you submit, you are required to create a folder with your name (Last name first, then First name), CS422, HW2, e.g., LastName_FirstName_CS422_HW2. Type your answers into a text file (**only `.txt`, `.doc`, and `.pdf` file formats are accepted**) and save it in this folder. Put all your Java programs (`*.java`) as well as output files in the same folder. Zip this folder, and submit it as one file to Desire2Learn. Do not hand in any printouts. Triple check your assignment before you submit. **If you submit multiple times, only your latest version will be graded and its timestamp will be used to determine whether a late penalty should be applied.**

Short Answers

P1. (10pts) In the following data set, the attribute $status$ is the class label. Calculate the information gains for attributes age and $salary$, repectively. If you are going to split the data set into smaller partitions, which of the two would you choose? Why?

department	*age*	*salary*	*status*
sales	31...40	41K...50K	senior
sales	<=30	<=40K	junior
sales	31...40	<=40K	junior
systems	<=30	41K...50K	junior
marketing	31...40	>50K	senior
systems	<=30	41K...50K	junior
systems	>40	>50K	senior
marketing	31...40	41K...50K	senior
marketing	31...40	41K...50K	junior
systems	>40	<=40K	senior
sales	<=30	<=40K	junior

P2. (9pts) For each of the measures: (a) sensitivity, (b) specificity, and (c) accuracy, provide an example in which the measure is most appropriate for evaluating the quality of classification. Explain your answer.

P3. (5pts) Consider two classifiers A and B. On one data set, a 10-fold cross validation shows that classifier A is better than B by 3%, with a standard deviation of 7% over 100 different folds. On the other data set, classifier B is better than classifier A by 1%, with a standard deviation of 0.1% over 100 different folds. Which classifier would you prefer on the basis of this evidence, and why?

P4. (5pts) Consider the rules $age > 40 \rightarrow yes$ and $age \leq 50 \rightarrow no$. Are these two rules mutually exclusive? Are these two rules exhaustive? Explain your answers.

Programming Questions

P5. (35pts) A high-level summary of the nearest-neighbor classification algorithm is shown as follows:

---

k-nearest neighbor classification algorithm

---

1. Let k be the number of nearest neighbors and D be the set of training samples.
2. **for** each test sample z **do**
3. Compute $d(z, x_i)$, the distance between z and every sample $x_i \in D$.
4. Select $D_z \subseteq D$, the set of k closest training samples (neighbors) to z.
5. Use weighted majority voting scheme to determine the class label for z.
6. **end for**

It computes the distance between each test sample z and all the training samples $x_i \in D$ $(i = 1,2,\cdots,|D|)$ to determine its nearest-neighbor list D_z. Once the list is obtained, the test sample is classified based on the mostly voted class of its nearest neighbors:

$$C_z = \underset{v}{\operatorname{argmax}} \sum_{\boldsymbol{x_i} \in D_z} w_i \times I(v = C_i)$$

where v is a class label, $w_i = 1/d(\boldsymbol{z}, \boldsymbol{x_i})^2$ (the weight of each class vote), C_i is the class label associated with $\boldsymbol{x_i}$ (a neighbor of z), and $I(\cdot)$ is an indicator function that returns 1 if the condition is true and 0 otherwise.

In this programming question, you will implement the k-NN classification algorithm and test your program on the UCI Glass Identification data set. The detailed information for this data set is available at https://archive.ics.uci.edu/ml/datasets/Glass+Identification. You are recommended to read the data description before you get started.

The Glass Identification data set contains 214 glass samples. Each sample has 11 attributes:

ID	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type

The first attribute is the ID number for this sample and the last attribute is the class label. Attributes 2~10 describe the properties of the sample. The distance is calculated based on these properties.

The data set has already been divided into two a training set (80% of the data, 171 samples) and a test set (20% of the data, 43 samples), stored in two text files "**`glassTraining.txt`**" and "**`glassTest.txt`**", respectively. The k-NN algorithm runs on the training set and makes the prediction for the samples in the test set.

a. Completing the `Homework2` class

You are provided with two files "**`Homework2.java`**" and "**`TestHomework2.java`**". You are required complete the following three methods in the former:

(Note: you should NOT change the contents of the training and test arrays in any of the following methods!)

`double[][] findNeighbors(double[][] trainingSet, double[] testSample, int k)`

This method takes as parameters a 2-D **`double`** array for training set (each row has 11 slots, with the first slot the ID of a sample and the last slot the class label of the sample), a 1-D **`double`** array for the test sample (only 10 slots included, does not include class label), and an integer **`k`** for the number of nearest neighbors. Here is an example of the **`testSample`** array (note that the last column, i.e., class label, was removed. You can compare it with the first sample in the test set):

`{3, 1.51618, 13.53, 3.55, 1.54, 72.99, 0.39, 7.78, 0, 0}`

You can call the **`euclidean`** method that is provided to you to obtain the **Euclidean distance** between the test sample and every training sample in order to obtain the k nearest neighbors for the test sample. **The distance measure should not include the first column, i.e., the ID number of the sample.** To arrange the neighbors of the test sample in ascending order (so that you can select the top k nearest neighbors), you can call **`selectionSort`** which sorts a 2-D **`double`** array based on the second column.

At the end of this method, you should return the neighbors in a 2-D **`int`** array, with first slot the neighbor ID's, the second slot the neighbor class labels, and the third slot the distances between the test sample and the neighbors.

`int weightedMajorityVote(double[][] neighbors)`

This method takes as parameter a 2-D **`double`** array which contains all neighbors with their associated class labels and distances. It calculates the weighted sum of votes for each class and returns the mostly voted class label.

`double measureAccuracy(double[][] trainingSet, double[][] testSet, int k)`

This method takes as parameters a 2-D **`double`** array for training set, a 2-D **`double`** array for test set, and an integer **`k`** for the number of neighbors. It calculates and returns the prediction accuracy (# of correctly predicted test samples divided by the total number of test samples) of the k-NN model on the test set. Make sure you remove the last column of the test sample when passing it into the **`findNeighbors`** method, so each test sample contains only 10 attributes!

Note that you are only supposed to touch the above methods. You are NOT allowed to create any other methods, instance variables, or make any changes to methods other than the above methods or files other than "`Homework2.java`". Points will be taken off if you fail to follow this rule.

b. Code Testing

You are provided with a test driver implemented by "**`TestHomework2.java`**" (**Do not make any changes to this file!**) so there is no need to write your own. You are also given two text files "**`glassTraining.txt`**" and "**`glassTest.txt`**" that contain training and test samples, respectively.

Depending on your programming environment, the data file might need to be placed in different folders so that you test driver can read it. For jGRASP, you can leave the data file in the same folder as your java files. For NetBeans, you should place it in your project folder in which you see directories like **`build`**, **`nbproject`**, and **`src`**, etc.

Once you have completed the required methods, you can run the test. You should create a plain text file named "**output.txt**", copy and paste the output (if your code crashes or does not compile, copy and paste the error messages) to this file and save it.

Grading Rubric:

Code does not compile: -10
Code compiles but crashes when executed: -5
Changes were made to things other than the required methods: -5
Has output file: 5
findNeighbors changes the content of the array parameters: -5
findNeighbors does not use Euclidean distance or calculates on wrong attributes: -5
weightedMajorityVote changes the content of the array parameter: -5
weightedMajorityVote does not use weight or uses incorrect weight: -5
measureAccuracy changes the content of the array parameters: -5
measureAccuracy does not remove the class label of test samples: -5
measureAccuracy does not call **findNeighbors** or **weightedMajorityVote**: -5
Code passes 10 test cases: 30 (each test case worth 3 points)

Sample Output:

```
Test 1: findNeighbors(testSample id = 3, k = 5) - [Passed]
 --------------
 Expected:
   ID        Class       Dist
  ..........................
   74          2        0.335
   82          2        0.433
    2          1        0.493
  123          2        0.497
   73          2        0.537

 Yours:
   ID        Class       Dist
  ..........................
   74          2        0.335
   82          2        0.433
    2          1        0.493
  123          2        0.497
   73          2        0.537

Test 2: findNeighbors(testSample id = 29, k = 3) - [Passed]
 --------------
 Expected:
   ID        Class       Dist
  ..........................
   15          1        0.208
   34          1        0.259
   31          1        0.274
```

```
 Yours:
   ID        Class       Dist
  ..........................
   15          1        0.208
   34          1        0.259
   31          1        0.274

...

Test 5: weightedMajorityVote - [Passed]
   ID        Class       Dist
  ..........................
    5          1        0.105
    6          5        0.113
    7          2        0.197
    8          4        0.212
 --------------
 Expected: 1
 Yours: 1

Test 6: weightedMajorityVote - [Passed]
   ID        Class       Dist
  ..........................
    9          3        0.143
   10          2        0.176
   11          1        0.263
   12          3        0.294
   13          5        0.312
   14          4        0.334
 --------------
 Expected: 3
 Yours: 3

...

Test 10: measureAccuracy(k = 20) - [Passed]
 --------------
 Expected: 0.674
 Yours: 0.674

Total test cases: 10
Correct: 10
Wrong: 0
```