CSE6242 / CX4242: Data and Visual Analytics | Georgia Tech | Fall 2015

# Homework 4: Decision Tree and Weka

Due: Monday, November 30, 2015, 11:55PM EST

Prepared by Meera Kamath, Gopi Krishnan, Siddharth Raja, Ramakrishnan Kannan, Akanksha, Polo Chau

## Submission Instructions:

It is important that you read the following instructions carefully and also those about the deliverables at the end of each question or you may lose points.

- ❏ Submit a single zipped file, called "HW4-{YOUR_LAST_NAME}-{YOUR_FIRST_NAME}.zip", containing all the deliverables including source code/scripts, data files, and readme. Example: 'HW4-Doe-John.zip' if your name is John Doe. Only .zip is allowed (no .rar, etc.)
- ❏ You may collaborate with other students on this assignment, but you must write your own code and give the explanations in your own words, and also mention the collaborators' names on T-Square's submission page. All GT students must observe the honor code. Suspected plagiarism and academic misconduct will be reported and directly handled by the Office of Student Integrity (OSI). Here are some examples similar to Prof. Jacob Eisenstein's NLP course page (grading policy):
  - ❏ **OK:** discuss concepts (e.g., how cross-validation works) and strategies (e.g., use hashmap instead of array)
  - ❏ **Not OK:** several students work on one master copy together (e.g., by dividing it up), sharing solutions, or using solution from previous years or from the web.
- ❏ If you use any "*slip days*", you must write down the number of days used in the T-square submission page. For example, "Slip days used: 1"
- ❏ At the end of this assignment, we have specified a folder structure about how to organize your files in a single zipped file. 5 points will be deducted for not following this strictly.
- ❏ Wherever you are asked to write down an explanation for the task you perform, please stay within the word limit or you may lose points.
- ❏ In your final zip file, please do not include any intermediate files you may have generated to work on the task, unless your script is absolutely dependent on it to get the final result (which it ideally should not be).

You will implement decision trees and learn to use Weka. We estimate Task 1 will take 10 hours and Task 2 will take 2 hours. These estimates can vary greatly, based on your computing background.

## Task 1: Decision Trees (70 points)

In this task, you will implement a well-known decision tree classifier. The performance of the classifier will be evaluated by 10-fold cross validation on a provided dataset. Decision trees and cross validation were covered in class (slides).

You will implement a decision tree classifier from scratch using either: (1) Python with the skeleton code we provide; or (2) other major programming languages of your choice (e.g., Java, C++, R). You must not use existing machine learning or decision tree libraries.

Download the wine dataset here, a dataset often used for evaluating classification algorithms, where the classification task is to determine whether a wine quality is over 7. We have mapped the wine quality scores for you to binary classes of 0 and 1. Wine scores from 0 to 6 (inclusive) are mapped to 0, wine scores of 7 and above are mapped to 1. You will be performing binary classification on the dataset. The dataset is extracted from the UCI machine learning repository https://archive.ics.uci.edu/ml/datasets/Wine+Quality . The data is stored in a comma separated file (csv). Each line describes a wine, using 12 columns: the first 11 describe the wine's characteristics (details), and the last column is a ground truth label for the quality of the wine (0/1). You must not use the last column as an input feature when you classify the data.

### A. Implementing Decision Tree (30 pt)

While implementing your decision tree, you will address the following challenges:

- Choosing the attribute for splitting (page 25-26 in the slides)
  - You can use entropy and information gain covered in the class.
- When to stop splitting (page 28)
  - Implementing advanced techniques, such as pruning or regularization (page 29), is **completely optional**.
  - You do not need to strictly follow the three "stopping" conditions on page 28; you may use similar (or better) approaches.

In your decision tree implementation, you may apply any variations that you like (e.g., using entropy, Gini index, or other measures; binary split or multi-way split). However, you must explain your approaches and their effects on the classification performance in a text file ***description.txt***.

We provide skeleton code here written in Python. It helps you set up the environment (loading the data and evaluating your model). You may choose to use this skeleton, write your own code from scratch in Python or other major languages (e.g., Java, C++, R).

## B. Evaluation using Cross Validation (20 pt)

You will evaluate your decision tree using 10-fold cross validation. Please see the lecture slides for details. In a nutshell, you will first make a split of the provided data into 10 parts. Then hold out 1 part as the test set and use the remaining 9 parts for training. Train your decision tree using the training set and use the trained decision tree to classify entries in the test set. Repeat this process for all 10 parts, so that each entry will be used as the test set exactly once. To get the final accuracy value, take the average of the 10 folds' accuracies.

With correct implementation of both parts (decision tree and cross validation), your classification accuracy should be around 0.78 or higher.

## C. Improvements (20 pt)

Try and improve your decision tree algorithm. Some examples of strategies are:

- Different splitting strategies
- Different stopping criteria
- Pruning tree after splitting
- Use randomness and/or bagging

Your improvement can be simple; using as few as one or two simple heuristics or ideas is acceptable. The goal here is for you to try different ways to improve the classifier. You do not need to implement separate code for this improvement. It is OK to build on top of your initial implementation and only submit the best (final) version of your decision tree (with your improvements integrated). But you should discuss what you have tried and why you think it performs better in *description.txt* .

In the text file, you will also report the performance comparison between the implementation you have tried. For example,

- Initial (in section A): 0.78
- After applying strategy 1: 0.89
- After applying strategy 2: 0.91

## Deliverables

1. **source code**: A source file(s) of your program. The source files should contain brief comments (what is this section for, e.g., calculating information gain). If you don't use the skeleton, include a **readme.txt** explaining how to compile and run the code on the provided data. It should be runnable using a single command via terminal.

2. **description.txt**: This file should include:
    a. How you implemented the initial tree (Section A) and why you chose your approaches (< 50 words)
    b. Accuracy result of your initial implementation (with cross validation)
    c. Explain improvements that you made (Section C) and why you think it works better (or worse) (< 50 words)
    d. Accuracy results for your improvements (It will be graded on a curve. Full credit (10 pt) will be given to students with relatively high accuracy.)

## Task 2: Using Weka (30 points)

You will use Weka to train classifiers for the same dataset used in Task 1, and compare the performance of your implementation with Weka's.

Download and install Weka. Note that Weka requires Java Runtime Environment (JRE) to run. We suggest you install the latest JRE, to avoid Java or runtime-related issues.

How to use Weka:
- Load data into *Weka Explorer*: Weka supports file formats such as arff, csv, xls.
- Preprocessing: you can view your data, select attributes, and apply filters.
- Classify: under *Classifier* you can select the different classifiers that Weka offers. You can adjust the input parameters to many of the models by clicking on the text to the right of the *Choose* button in the Classifier section.

These are just some fundamentals of Weka usage. You can find many tutorials on how to use weka on the Internet.

### A. Experiment (15 pt)

Run the following experiments. After each experiment, report your **parameters, running time, confusion matrix, and prediction accuracy**. An example is provided later, under the "Deliverables" section.

1. Decision Tree - C4.5 (J48) is commonly used and similar to what you implemented in Task 1. Under *classifiers -> trees*, select J48. For the Test options, choose **10-fold cross validation**, which should be the same for A2 and A3. (5 pt)
2. Support Vector Machines - Under the *classifiers -> functions* select SMO. (5 pt)
3. Your choice -- choose any classifier you like from the numerous classifiers Weka provides. You can use package manager to install the ones you need. (5 pt)

## B. Discussion (15 pt)

Discuss in your report and answer following questions:

1. Compare the Decision Tree result from A1 to your implementation in Task 1 and discuss possible reasons for the difference in performance. (< 50 words, 5 pt)
2. Describe the classifier you choose in A3: what it is, how it works, what are its strengths & weaknesses. (< 50 words, 5 pt)
3. Compare and explain three approaches' classification results in Section A, specifically their running times, accuracies, confusion matrices. If you change any of the parameters, briefly explain what you have changed and why they improve the prediction accuracy. (< 100 words, 5 pt)

## Deliverables

**report.txt** - a text file containing the Weka result and your discussion for all questions above. For example:

```
Section A
1.
        J48 -C 0.25 -M 2
        Time taken to build model: 3.73 seconds
        Overall accuracy: 86.0675 %
        Confusion Matrix:
            a     b   <-- classified as
         33273  2079 |     a = no
          4401  6757 |     b = yes
2.
…

Section B
1. The result of Weka is 86.1% compared to my result <accuracy> because…
2. I choose <classifier> which is <algorithm>…
...
```

## Submission Guidelines

Submit the deliverables as a single <u>**zip**</u> file named **hw4-*Lastname-Firstname.zip*** (should start with lowercase hw4). Write down the name(s) of any students you have collaborated with on this assignment, using the text box on the T-Square submission page.

The zip file's directory structure must exactly be (when unzipped):

```
Task1/
     description.txt
     tree.py
     OR
     tree.xxx
     additional_source_files (if you have)
     readme.txt (unless you used tree.py)

Task2/
     report.txt
```

You must follow the naming convention specified above.