CSE6242 / CX4242: Data and Visual Analytics | Georgia Tech | Spring 2016

Due: Friday, April 22, 2016, 11:55PM EST

Prepared by Gopi Krishnan Nambiar, Nilaksh Das, Pradeep Vairamani, Ajitesh Jain, Vishakha Singh, 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". Each slip day equals 24 hours. E.g., if a submission is late for 30 hours, that counts as 2 slip days.
- ❏ At the end of this assignment, we have specified a folder structure of 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, stay within the word limit or you may lose points.
- ❏ In your final zip file, 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).
- ❏ After all slip days are used up, 5% deduction for every 24 hours of delay. (e.g., 5 points for a 100-point homework)
- ❏ We will not consider late submission of any missing parts of an homework assignment or project deliverable. To make sure you have submitted everything, download your submitted files to double check.

Task 0: Download HW skeleton

Download the HW skeleton from this link that contains starter code for Task 1.

Task 1: Random Forest (70 points)

In this task, you will implement a random forest classifier in Python. The performance of the classifier will be evaluated using 10-fold cross validation on a provided dataset. For details on random forest, see Chapter 15 in the "Elements of Statistical Learning" book and lecture slides. You **must not** use existing machine learning or random forest libraries. We have also set up an **in-class Kaggle competition** in which your classifier will compete with the rest of the class!

To train your classifier, you will be working on the wine dataset, which is often used for evaluating classification algorithms. The classification task is to determine whether the quality of a particular wine 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 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 (see details on Data page for the competition on Kaggle), 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 Random Forest (40 pt)

The main parameters in a random forest are:

1. Which attributes of the whole set of attributes do you select to find a split?
2. When do you stop splitting leaf nodes? You may even choose to use decision stumps which are just 1 node deep.
3. How many trees should be in the forest?

In your 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 have prepared starter code written in Python (*RandomForest.py*) which you will be using. This would help you setup the environment (loading the data and evaluating your model). For students new to Python, we have also created reference documentation that might help you get started with the language.

B. Evaluation using Cross Validation (30 pt)

You will evaluate your random forest (model) using 10-fold cross validation (also, lecture slide 13-15). In the nutshell, you will first divide 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 model using the training set and use the trained model to classify entries in the test set. Repeat this process for all 10 parts, so that each entry will be used for testing exactly once. To compute your model's final accuracy, take the average of the 10 folds' accuracies. With correct implementation of both parts (random forest and cross validation), your classification accuracy should be above 0.786. Solutions achieving accuracy less than this may receive some penalty.

Note: In random forests, evaluation using cross validation is not necessary (see slide 9 of class slides on Ensemble Methods). However, we would like you to learn how to implement and apply cross validation in practice, as it is a fundamental model evaluation approach.

C. Kaggle competition (15 pt bonus)

Kaggle is a platform for predictive modelling and analytics competitions. You can submit the predictions of your random forest on a test dataset at the competition page hosted on this platform. **The top submissions will receive significant amount of bonus points**. The starter code is accompanied by a tester file (*CSE6242HW4Tester.pyc*) that automatically evaluates your random forest implementation and creates a Kaggle submission file every time you run your code. The generated submission file is named *cse6242_hw4_submission_LastName-FirstName.csv*. You MUST NOT change this file in any way; submit this file to Kaggle.

Sign up on Kaggle with your @gatech.edu email address and use your Georgia Tech account name (e.g. gpburdell3) **as both your username and display name**. If you have already registered with Kaggle using your GaTech email address, please make a private Piazza post to instructors with the title "HW4 Kaggle Registration Details" and mention your existing Kaggle **username**, **display name** and your **Georgia Tech account name**.

Note that Kaggle has 2 leaderboards, a public one that is visible to all students, and a private one only visible to the instructors. Each leaderboard use a different dataset to rank submissions. **We will award bonus points based on your model's performance on the private dataset/leaderboard.** Using two similar, independent datasets/leaderboards is a common practice, which prevents competitors from overfitting their models to the public dataset. You can read more about this here.

Bonus points will be awarded as follows:
Rank 1 to 5: 15 pt
Rank 6 to 25: 10 pt
Rank 26 to 75: 6 pt
Rank 76 and above: 3 pt

You may use advanced heuristics to improve the performance of your random forest, which you should mention in **description.txt**. It is OK to build on top of your initial implementation and only submit the best (final) version of your random forest. Also mention the final test accuracy of your implementation (e.g., after tuning parameters of your random forest using cross-validation).

Deliverables

1. **RandomForest.py**: The source file of your program. This should also contain brief comments mentioning what each section of the code does, e.g., calculating information gain.

2. **cse6242_hw4_submission_LastName-FirstName.csv**: The submission file generated by the tester file.

3. **description.txt**: This file should include:
    a. How you implemented the initial random forest (Section A) and why you chose your specific approach (< 75 words)
    b. Accuracy of your implementation (with cross validation)
    c. Explain improvements that you made (Section C) and why you think it works better (or worse) (< 50 words)

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. There are plenty of online tutorials.

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 below, under the "Deliverables" section. For the Test options, choose **10-fold cross validation**

1. **Random Forest**. Under *classifiers -> trees*, select RandomForest. You might have to preprocess the data before using this classifier. (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)

1. Compare the Random Forest 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; its strengths & weaknesses. (< 50 words, 5 pt)
3. Compare and explain the three approaches' classification results in Section A, specifically their running times, accuracies, and confusion matrices. If you have changed/tuned any of the parameters, briefly explain what you have done 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):

```
hw4-LastName-FirstName/

    Task1/
        cse6242_hw4_submission_LastName-FirstName.csv
        description.txt
        hw4-data.csv
        RandomForest.py
        CSE6242HW4Tester.pyc

    Task2/
        report.txt
```

You must follow the naming convention specified above.