CS 165B – Machine Learning, Spring 2018

Assignment #3
Due Thursday, May 24 by 10:00am PST

This homework has 2 parts!

- **a.** Written assignment to be turned into Gauchospace
- **b.** Coding assignment to be submitted to CodaLab (Directions to follow)

Notes:

- **This assignment is to be done individually**. You may discuss the problems at a general level with others in the class (e.g., about the concepts underlying the question, or what lecture or reading material may be relevant), but the work you turn in must be solely your own.
- Be aware of the late policy in the course syllabus – i.e., you only have four late days for all the assignments,
- Justify every answer you give – **show the work** that achieves the answer or **explain** your response.
- Any updates or corrections will be posted on the Assignments page (of the course web site), so check there occasionally.
- Be sure to **include your name and perm number on the required place of the assignment (at the top of the 2nd page).**
- Be sure to **strictly stick to the format provided for answering all the questions**.
- **All assignments must be clear and legible.** It is recommended to type the solutions on this PDF directly. If you'll be submitting a handwritten assignment, please ensure that it's readable and neat. If your writing is not easily readable, your solution is not easy to follow on the page, or your PDF is not of very high quality, your assignment will not be graded. DO NOT submit picture of your written work. (If you must scan your written work in, use a high-quality scanner. Plan in advance.)
- Be sure to re-read the "Academic Integrity" on the course syllabus. You must complete the section below. If you answered Yes to either of the following two questions, give corresponding full details.

Did you receive any help whatsoever from anyone in solving this assignment? Yes No

Did you give any help whatsoever to anyone in solving this assignment? Yes No

---

Name:

Perm Number:

All problems use the following features to learn the concept *GoodMovie*, which predicts whether a movie is good or not based on several attributes:

> Movie's budget = { Low, Medium, High }
> Genre = { Documentary, Drama, Comedy }
> Famous actors = { No, Yes }
> Director = { Unknown, Great }

The training data for *GoodMovie* is:

#	*Budget*	*Genre*	*FamousActors*	*Director*	*GoodMovie*
1	Low	Documentary	Yes	Unknown	No
2	Low	Documentary	Yes	Great	No
3	Medium	Documentary	Yes	Unknown	Yes
4	High	Drama	Yes	Unknown	Yes
5	High	Comedy	No	Unknown	No
6	High	Comedy	No	Great	Yes
7	Medium	Comedy	No	Great	No
8	Medium	Documentary	No	Unknown	No
9	Low	Comedy	Yes	Great	Yes
10	Low	Drama	Yes	Unknown	No
11	Low	Comedy	No	Unknown	No
12	High	Drama	No	Unknown	Yes
13	Low	Drama	No	Great	Yes
14	Medium	Drama	Yes	Great	Yes
15	Medium	Documentary	No	Unknown	Yes
16	High	Drama	Yes	Great	Yes

Problem #1 [12 points]
In a conjunctive hypothesis space learning approach applied to the *GoodMovie* scenario above (basic CHS, not including disjunctions):

(a) How many possible (conjunctive) hypotheses are there?

(b) What is the least general hypothesis for *GoodMovie* after observing training data points 13 and 16 (only)?

(c) What is the most general hypothesis for *GoodMovie* after observing training data points 13, 16, and 7 (only)?

(d) What is the least general hypothesis for *GoodMovie* after observing training data points 10, 11, and 13 (only)?

Problem #2 [20 points]

Based on the GrowTree and BestSplit-Class algorithms and using the entropy impurity function, create a decision tree to learn the *GoodMovie* concept. Show how each node is decided (based on comparing impurity measures), then draw the full decision tree.

If there are ties in impurity measures, give higher priority to attributes and values according to their order on page 1: that is, Budget is the highest priority feature and Director is the lowest priority; within Budget, Low is the highest priority value, followed by Medium and then High. [Normally these might be randomly chosen, but we'll use this "inductive bias."] If there are ties to determine a leaf's label, give priority to "Yes."

Now apply this learned concept (decision tree) to the three test data sets posted on the Assignments page, and list the correctly and incorrectly classified examples (by number), for each test data set.

What is the error rate of the decision tree on the training data? On each test data set?

Problem #3 [15 points]
From your decision tree in problem 2, create a ranking tree based on empirical probabilities of the leaves, using Laplace correction. Draw the tree and label each leaf with its estimated *GoodMovie* probability and its ranking.

Compute the error rate and the accuracy of applying the ranking tree to the training data and to the test data files posted on the Assignments page.

Problem #4 [28 points]
Write a **Python2** program called **`hw3.py`** that creates a decision tree classifier for the *GoodMovie* problem using the **scikit-learn** function **`DecisionTreeClassifier`**. The training and testing data sets are available on the Assignments page of the course web site. The file key.txt shows how features are represented as numeric values. (I.e., Budget=0 means Low, Budget=1 means Medium, etc.).

The syntax of your program should be:

```
% def run_train_test (training_file, testing_file)
```

The **`training_file`** and **`testing_file`** are file objects returnd by open() in Python. Basically, you need to do 3 things: examine the txt files and convert them into the desired input format, run DecisionTreeClassifier, calculate the results.

Be sure to see the scikit-learn documentation (at scikit-learn.org) for DecisionTreeClassifier. You'll need to use the **`fit`** and **`predict`** methods. Use all of the class defaults, including the Gini function for the impurity measure, except to **FIX the random_state to 0** (otherwise your output may not be consistent in each run, you can play around with it if you like but make sure you submit the results with it FIXED!)

Now modify your *GoodMovie* classifier to use **entropy** instead of Gini to measure impurity. (This merely involves setting the *criterion* parameter of DecisionTreeClassifier.) Run it on the same training and testing sets (Be sure to notice if the change in impurity measure modified the results or not!)

As output, the program should print out the following numbers: the number of true positives, true negatives, false positives, and false negatives, and the error rate, for the two classifiers respectively, as shown here: (keys should match the exact strings!)

```
{
        "gini": {
                "True positives" = 100, "True negatives" = 100,
                "False positives" = 100, "False negatives"= 100,
                "Error rate" = 0.99
                }
        "entropy": {
                "True positives" = 100, "True negatives" = 100,
                "False positives" = 100, "False negatives"= 100,
                "Error rate" = 0.99
                }
}
```

(Note: These numbers are made up, for purposes of illustration only.)
You can test your program with the training and testing sets provided on the Piazza.

CodaLab

Submission Instructions

You will need to create a zip file containing your submission. You should use the following command:

```
% zip -r submission.zip hw3.py
```

The **submission.zip** file is what will be uploaded to CodaLab where it is automatically tested. **DO NOT** put your code in a new directory, as our testing functions will not be able to import your code if you do.

CodaLab Competition Link

You will submit your code through this competition and get results for your homework. Below is the competition where you should submit

https://competitions.codalab.org/competitions/19072?secret_key=482b80e3-6739-4345-9879-8a11fb9fce11

Registration Instructions

Please follow the directions here to register and participate in our class competition: https://github.com/codalab/codalab-competitions/wiki/User_Participating-in-a-Competition

****Register with you UMail account!****

Remember to register a CodaLab Competitions account using your umail account so that the username will be your UCSBNetID. After that, log into CodaLab Competitions and set up your team name (whatever nickname you like). To protect your privacy, only the team names will be shown on the leader-board and your usernames will be anonymous. After your submission finishes running, please choose to submit it to the leader-board. Note that here team name is equivalent to your nickname, and it is still an independent homework assignment. You must implement the four algorithms according to the instructions above.