CS373 HW3

April 14, 2018

Due date: 11:59pm Sunday April 01, 2018.

Instructions for submission:

You need to implement this assignment from scratch in Python. Do not use any already implemented models like those in the scikit-learn library. Also, programs that print more than what is required will be penalized.

Your code needs to run in the server: `data.cs.purdue.edu`. Check that it works there by executing your script from the terminal.

For part 1, submit your code as a Python program. For part 2, type your answers in a PDF file. As usual, label the plots with the question number. Your homework **must** contain your name and Purdue ID at the start of the file. If you omit your name or the question numbers for the plots, you will be penalized.

To submit your assignment, log into `data.cs.purdue.edu` (physically go to the lab or use ssh remotely) and follow these steps:

1. Make a directory named `yourusername-hw3` (all letters in lower case) and copy your PDF file and Python file inside it. To do it remotely, use:

   ```
   scp ./path/to/your-file.pdf your-id@data.cs.purdue.edu:./remote/path/from-home-dir/
   ```

2. Go to the directory containing `yourusername-hw3` (e.g., if the files are in /homes/dan/dan-hw3, go to /homes/dan), and execute the following command:

   ```
   turnin -c s373 -p hw3 yourusername-hw3
   ```

   (e.g. Dan would use: `turnin -c s373 -p hw3 dan-hw3` to submit his work)

   Note that `s373` is the course name for turnin. **It is not a typo.**

3. To overwrite an old submission, simply execute this command again.

4. To verify the contents of your submission, execute the following command:

   ```
   turnin -v -c s373 -p hw3
   ```

1 Specification

1.1 Dataset Details

Download the files `adult.data` and `adult.test` from the course page.

In order to read the CSV data and obtain a numpy matrix X with the necessary attributes you can use the following code:

```
import pandas as pd
data = pd.read_csv(filename, sep=', ', quotechar='"', header=None, engine='python')
X = data.as_matrix()
```

1.2 Code Details

Your python script should take the following arguments:

1. trainingFile: path to the training set.
2. testFile: path to the test set.
3. model: model that you want to use. In this case, we will use:
    - `vanilla` for the full decision tree
    - `depth` for the decision tree with static depth
    - `prune` for the decision tree with post-pruning
4. training set size as a percentage.

Each case may have some additional command-line arguments, which will be mentioned in its format section. Use `sys.argv` to get the list of arguments.

Your code should read the training set from trainingFile, extract the required features, train your decision tree on the training set, and test it on the test set from testFile. Name your file `decisiontree.py`.

For debugging purposes, you can use a small fraction of the dataset, for example, by using `X[:1000]` to work with the first 1000 data points.

2 Part 1: Decision Trees

Note: You need to submit only your code as a separate file (`decisiontree.py`) for this section.

Features: Consider the first 8 attributes as the input features: {workclass, education, marital-status, occupation, relationship, race, sex, native-country}.

Class label: Consider the last attribute `salaryLevel` as the class label. This takes the values `>50k` and `<=50k`.

1. Implement a binary decision tree with no pruning using the ID3 algorithm. **(20 points)**

Format:

```
$ python decisiontree.py ./path/to/file1.csv ./path/to/file2.csv vanilla 80
Training set accuracy: 0.9123
Test set accuracy: 0.8123
```

The fourth argument is the training set percentage. So, for example, that above command means that we use only the first 80% of the training data from file1. (We use all of the test data from file2.)

2. Implement a binary decision tree with a given maximum depth. **(25 points)**

   Format:

   ```
   $ python decisiontree.py ./path/to/file1.csv ./path/to/file2.csv depth 50 40 14
   Training set accuracy: 0.9123
   Validation set accuracy: 0.8523
   Test set accuracy: 0.8123
   ```

   The fourth argument is the training set percentage and the fifth argument is the validation set percentage. The sixth argument is the value of maximum depth.

   So, for example, the above command would get a training set from the first 50% of file1 and get a validation set from the last 40% of file1 (the two numbers need not add up to 100% because we sometimes use less training data). Finally, we set the maximum depth of the decision tree as 14. As before, we get the full test set from file2.

   Note that you have to print the validation set accuracy for this case.

3. Implement a binary decision tree with post-pruning using reduced error pruning. **(30 points)**

   ```
   $ python decisiontree.py ./path/to/file1.csv ./path/to/file2.csv prune 50 40
   Training set accuracy: 0.9123
   Test set accuracy: 0.8123
   ```

   The fourth argument is the training set percentage and the fifth argument is the validation set percentage.

   So, for example, the above command would get a training set from the first 50% of file1 and get a validation set from the last 40% of file1. As before, we get the full test set from file2.

3 Part 2: Analysis

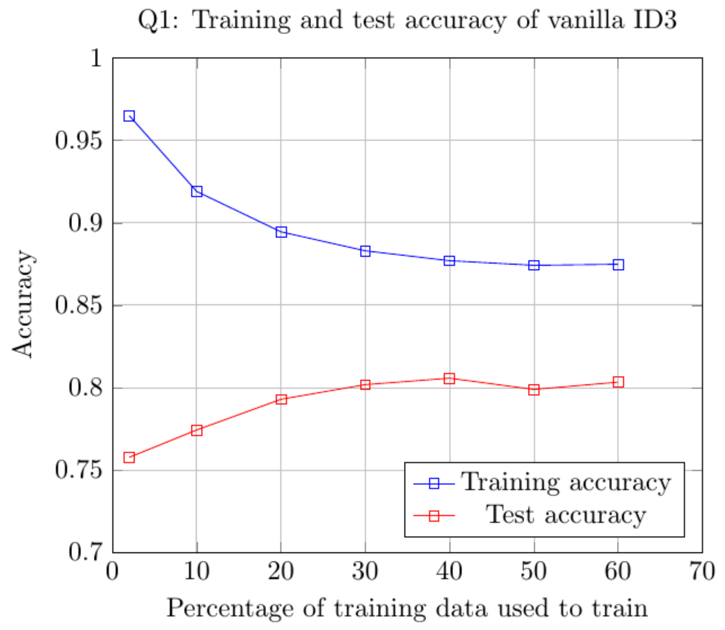
Note: You need to submit only your answers in the PDF for this section.

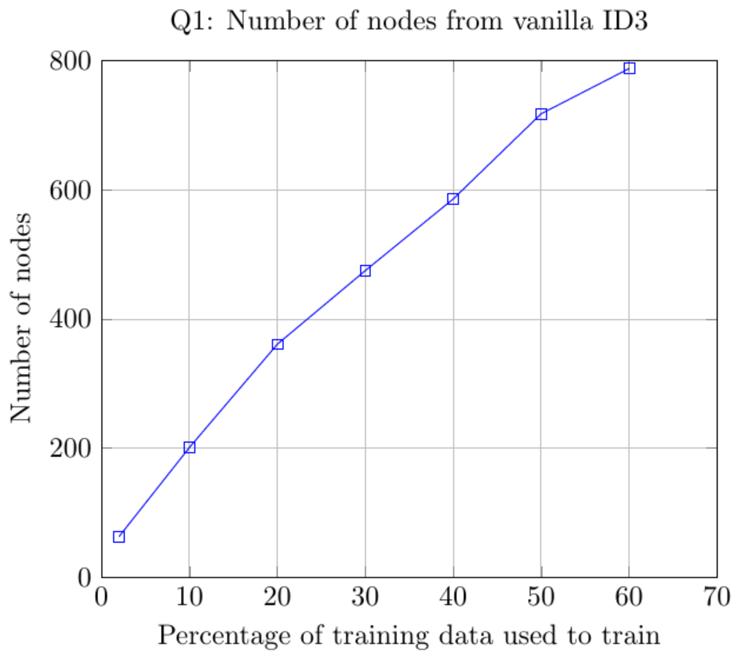
For the following questions, use `adult.data` as the training file and `adult.test` as the test file.

1. For the full decision tree (`vanilla`), measure the impact of training set size on the accuracy and size of the tree. (**4 points**)

Consider training set percentages {2%, 10%, 20%, 30%, 40%, 50%, 60%}

Plot a graph of test set accuracy and training set accuracy against training set percentage on the same plot. Plot another graph of number of nodes vs training set percentage.





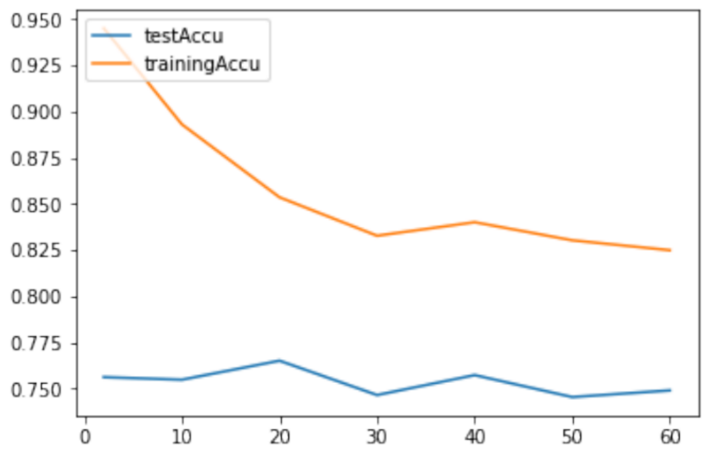
2. Repeat the same analysis for the static-depth case (`depth`). (**7 points**)

   Again, consider values of training set percentage from {2%, 10%, 20%, 30%, 40%, 50%, 60%}. The validation set percentage will remain 40% for all the cases.

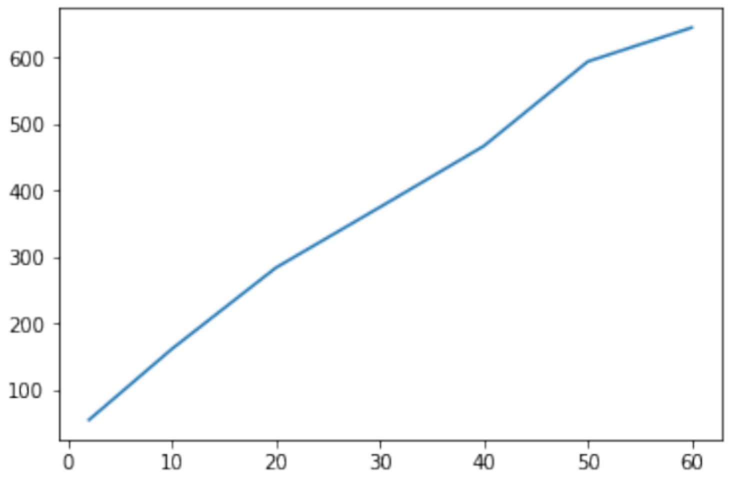
Consider values of maximum depth from {1, 4, 7, ..., 34} and pick the best value using the validation set accuracy. The accuracies you report will be the ones for this value of maximum depth. So, for example, if the best value of maximum depth for training set 10% is 4, you will report accuracies for 10% using 4; if for 20% it is 16, you will report accuracies for 20% using 16.

Plot a graph of test set accuracy and training set accuracy against training set percentage on the same plot. Plot another graph of number of nodes vs training set percentage.

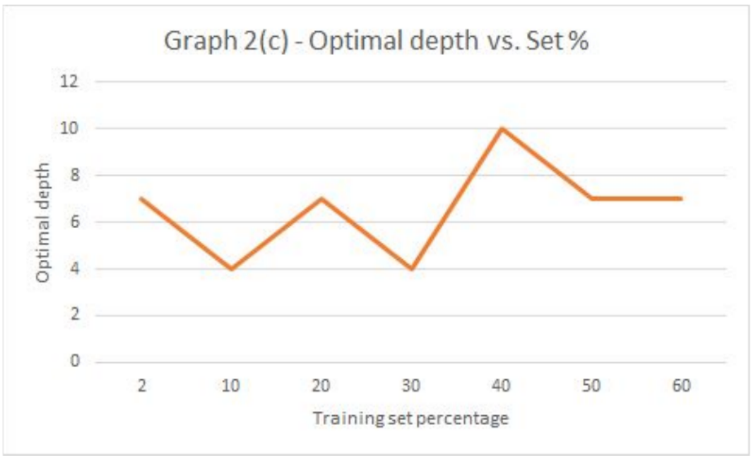
Finally, plot the optimal choice of depth against the training set percentage.



.



.

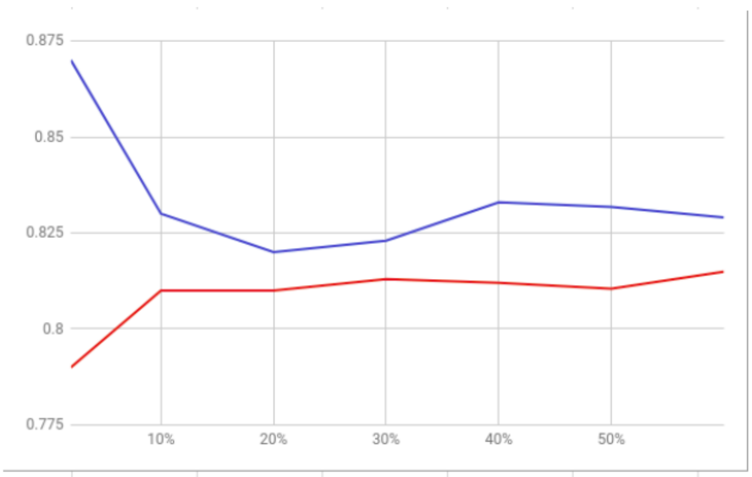


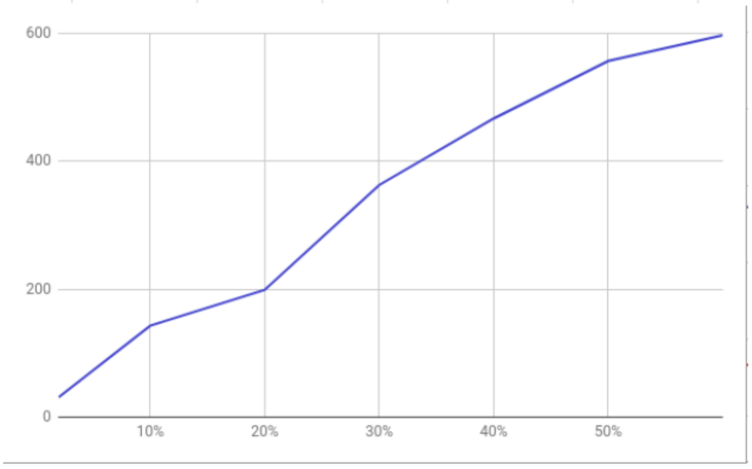
.

3. Repeat the above analysis for the pruning case (`prune`). (**7 points**)

   Again, consider values of training set percentage from {2%, 10%, 20%, 30%, 40%, 50%, 60%}. The validation set percentage will remain 40% for all the cases. You will use the validation set when deciding to prune.

   Plot a graph of test set accuracy and training set accuracy against training set percentage on the same plot. Plot another graph of number of nodes vs training set percentage.





.

4. Why don't we prune directly on the test set? Why do we use a separate validation set? (**3 points**)

   The test set is meant to represent unseen data. If we prune using it, then we would inadvertently fit our model to it and thus won't get an accurate measure of its performance on unseen data.

   (If they don't give a reason and just repeat the question, then they shouldn't get any points.

   For example, this doesn't give a reason: "Since we are building a model of the hypothesis set, we use the training set. If we were to do an experiment it would be more favourable to use the test set.")

5. How would you convert your decision tree (in the `depth` and `prune` cases) from a classification model to a ranking model? (**4 points**)

   That is, how would you output a ranking over the possible class labels instead of a single class label?

   Keep track of the possible class labels instead of just the majority vote and output them in the order of frequency.