CS 373 Fall 2019
Instructor: Romila Pradhan
I collaborated with (...). I affirm that I wrote the solutions in my own words and that I understand the solutions I am submitting.

CS373 Homework 3

Due date: Wednesday, October 30, 11:59pm

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 using python3. **For part 1, submit your code as a single file (ID3.py). For part 2 and 3, type your answers in a single PDF file. Compressed into zip file in the following format and upload it to the server.**

Your file structure must look like the following, otherwise you will loose points:

```
yourusername-hw3/
├── ID3.py
├── yourusername-hw3.pdf
└── *readme*
```

where you can include the `readme` if your code has special issues that we should be aware of. As usual, label the plots with the question number. Your homework **must** contain your **name** and **PUID** at the start of the file. If you omit your name or the question numbers for the plots, you will be penalized. We will also run through **code plagiarism checker** for your solutions, including all the previous years' codes
To submit your codes, log into `data.cs.purdue.edu` (physically go to the lab or use ssh remotely) and follow these steps:

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

   ```
   turnin -c cs373 -p hw3 yourusername-hw3.zip
   ```

   (e.g. Romila would use: `turnin -c cs373 -p hw3 romila-hw3.zip` to submit her work) Note that `cs373` is the course name for turnin.

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

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

   ```
   turnin -v -c cs373 -p hw3
   ```

Part 0 Specification

Install Python3.6.8 if Necessary

Note: If you are coding on `data.cs.purdue.edu` or already installed `python3` for the last homework, you may skip this step. Just run your program using `python3`.

You can install anaconda to configure the `Python` environment for Mac/Win/Linux at `https://www.anaconda.com/distribution/#download-section`. Make sure you install the right version. Then install the related packages, type this command in your terminal:

```
conda install pandas numpy scipy matplotlib
```

If you are not fully familiar with Python language, we recommend you to go though online tutorials before doing your homework. Recommended websites for you: `https://www.programiz.com/python-programming/tutorial` and `https://www.codecademy.com/learn/learn-python`.
You can also try out `Jupyter Notebook`[1] for real-time coding, which is quite similar to `R`.

Dataset Details

Find these files:

```
adult.data, adult.test
```

in the `HW3.zip` file.

We will consider the last attribute in the data `salaryLevel` as the class value, and the remaining 8 attributes in the dataset are the input features for the decision tree.

Input format

Your python script should take the following arguments:

1. `train-file`: path to the training set file. (`adult.data`)
2. `test-file`: path to the test set file. (`adult.test`)
3. `model`: model that you want to use. In this case, we will use:
   - `vanilla`: the full decision tree.
   - `prune`: the decision tree with post-pruning.
4. `train-set-size`: percentage of dataset used for training.

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

[1] `https://jupyter.org/install`

```
import sys
for x in sys.argv:
     print('arg: ', x)
```

Your code should read the training set from `train-file`, extract the required features, train your decision tree on the training set, and test it on the test set from `test-file`. Name your file `ID3.py`.
For debugging purposes, you can use a small fraction of the dataset, for example, by using `X[:100]` to work with the first `100` data points.

Model Details

Entropy

Entropy of a dataset is calculated over fraction of examples belonging to each class. For a set of examples S (with k class labels), if p_i denotes the fraction of examples that belong to the i-th class label {i=1, . . . , n}, then entropy of S is given by:

$$Entropy(S) = -\sum_{i=1}^{k} p_i log(p_i) \tag{1}$$

Information gain

We provide the formula for Information Gain where S_v is the set of examples where value of attribute A=v.

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_V|}{|S|} Entropy(S_v) \tag{2}$$

Note: The tree building function and prune function should be implemented in a recursive manner, otherwise your solution score will be penalized by 80%.

Part 1 Decision Trees

Note: You need to finish only your code as a separate file (ID3.py) for this section.
Your code should be optimized with numpy and pandas functions, and should not take an unreasonable amount of time. However, there is no strict 5 minute time limit for thie assignment. We will verify your training and testing modules, and you should not include your pre-trained model. Note that the numbers provided are not accurate and are just examples.

1. (**20 points**) Implement a binary decision tree with no pruning using the ID3 (Iterative Dichotomiser 3) algorithm[2]. Remember that last attribute `salaryLevel` is the class value and the remaining 8 attributes in the data file are the input features for the decision tree.

   Format of calling the function and accuracy you will get after training:

   ```
   $ python3 ID3.py train-file test-file vanilla 80
   Train set accuracy: 0.9123
   Test set accuracy: 0.8123
   ```

   The first and second argument is the filname of the train and test file. The fourth argument (80) is the training set percentage. The above example command means we use only the first 80% of the training data from `train-file`. (We use all of the test data from `test-file`.)

2. (**20 points**) Implement a binary decision tree with post-pruning using reduced error pruning. Remember that the last attribute `salaryLevel` is the class value and the remaining 8 attributes in the data file are the input features for the decision tree.
   Format of calling the function and accuracy you will get after training:

   ```
   $ python3 ID3.py train-file test-file prune 50 40
   Train set accuracy: 0.9123
   Test set accuracy: 0.8123
   ```

   The fourth argument (50) is the training set percentage and the fifth argument (40) is the validation set percentage.

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

[2]`https://en.wikipedia.org/wiki/ID3_algorithm`

Part 2 Analysis

Note: You need to submit only your answers in the PDF for this section.
It is recommended to use `numpy, matplotlib, seaborn` for plotting the learning curves. Do not submit the code used. Make sure your `xaxis, yaxis, title` is labeled appropriately.

1. (**7 points**) For the full decision tree (`vanilla`), measure the impact of training set size on the accuracy and size of the tree.
   Consider training set percentages $\{40\%, 60\%, 80\%, 100\%\}$.

   - Plot a graph of test set accuracy and training set accuracy against training set percentage on the same plot. Also include a line for the baseline default error that would be achieved if you just predicted the most frequent class label in the overall data (100% of the train data).
   - Plot another graph of number of nodes vs training set percentage.
   - Discuss the results (e.g. how is accuracy impacted by training set size and how it compares to just predicting the dominant class) in a few sentences. For each training set size, is the decision tree overfitting?

2. (**8 points**) Repeat the above analysis for the pruning case (`prune`).

   - Again, consider values of training set percentage: $\{40\%, 50\%, 60\%, 70\%, 80\%\}$. The validation set percentage will remain 20% 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. Also include a line for the baseline default error that would be achieved if you just predicted the most frequent class label in the overall data (100% of the train data).
   - Plot another graph of number of nodes vs training set percentage.
   - Discuss the results (e.g. how is accuracy impacted by training set size and how it compares to just predicting the dominant class) in a few sentences. For each training set size, is the decision tree overfitting?

Part 3 Theoretical questions

Note: You should not need to do any coding in order to answer these questions. Please answer the following questions in a few short sentences. Include the answers in the same PDF as Part 2, but label the section for Part 3.

1. (**2 points**) Will ID3 always include all the attributes in the tree?
2. (**2 points**) Why do we not prune directly on the test set and use use a separate validation set instead?
3. (**3 points**) How would you handle missing values in some attributes? Answer for both during training and testing.
4. (**4 points**) How would you convert your decision tree from a classification model to a ranking model (i.e., how would you output a ranking over the possible class labels instead of a single class label)?
5. (**4 points**) Consider the case where instead of binary values, the class label has continuous values. How would you adapt your decision tree to predict the class value of a test instance? (4 points)

Part 4 Extra Credit: Max Depth

Note: This part is completely optional. If you do complete it, it provides 10% extra credit. The code for this part must run in under 5 minutes on data.cs.purdue.edu. Include the answers in the same PDF as before, but add a section for Part 4.

1. Implement a binary decision tree with no pruning using the ID3 (Iterative Dichotomiser 3) algorithm[3]. Remember that last attribute `salaryLevel` is the class value and the remaining 8 attributes in the data file are the input features for the decision tree.

   In this tree, we will have a maxDepth parameter, which the tree's depth cannot go past this maxDepth parameter.

   Format of calling the function and accuracy you will get after training:

   ```
   $ python3 ID3.py train-file test-file maxDepth 50 40 5
   Train set accuracy: 0.9123
   Test set accuracy: 0.8123
   ```

   The fourth argument (`50`) is the training set percentage and the fifth argument (`40`) is the validation set percentage. The sixth argument (5) is the value of the maxDepth of the tree. The tree you build must not exceed this depth.

   So, for example, the above command would get a training set from the first 50% of `train-file` and get a validation set from the last 40% of `train-file`. As before, we get the full test set from `test-file`.) The depth of the tree would be at max 5.

2. Tune your maxDepth value on the validation set. Briefly discuss how you did this and what maxDepth value you selected. Plot a graph of at least 10 maxDepth values you tried on the validation set (including the one you selected), and compare with the performance on the validation set.

[3] `https://en.wikipedia.org/wiki/ID3_algorithm`