CS373 Homework 2

Due date: Oct 21th 11:59PM, 2020

Instructions for submission:

You need to implement this assignment from Our starter code using `Python3`. 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 single file (`yourusername-ID3.py`). You should use `turnin` to submit the code to the server. For part 2, type your answers in a single PDF file and submit to Gradescope**.

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-ID3.py` (e.g., if the files are in `/homes/dan/`, go to `/homes/dan`), and execute the following command:

   ```
   turnin -c cs373 -p hw2 yourusername-ID3.py
   ```

   (e.g. Dan would use: `turnin -c cs373 -p hw2 dan-ID3.py` to submit his 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 hw2
   ```

Part 0 Specification

You can use the default Python3 in the `data.cs.purdue.edu`. If you want to install a Python environment locally, check the instructions below.

Install Python 3.6.9

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, `Python-3.6.9-64-bit` installer is preferred. 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 `Pycharm IDE` (`https://www.jetbrains.com/pycharm/`) for coding debugging.

Dataset Details

in the `HW2.zip` file, it contains the following dataset files:

```
titanic-train.data, titanic-train.label, titanic-test.data, titanic-test.label
```

The overall attributes of dataset is shown in Table 2. The meaning of every feature is shown below:

Table 1: Feature meaning definition.

Feature Name	Definition
Survived	dead or survive (Bool);
Pclass	Class of Travel (Int);
Name	name of passenger(String);
Sex	Gender(Bool);
Age	Age of Passengers(Int);
Relatives	Number of relatives (Int);
IsAlone	If he/she has no relatives (Bool);
Ticket	ticket number (String);
Fare	Passenger fare (Float);
Embarked	Port of Embarkation (Int).
	C = Cherbourg (0), Q = Queenstown(0), S = Southampton(77);

Table 2: Samples in titanic raw data.

Survived	Pclass	Name	Sex	Age	Relatives	IsAlone	Ticket	Fare	Embarked
0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	1	Cumings, Mrs. John Bradley	female	38.0	1	0	PC 17599	71.2833	C
1	3	Heikkinen, Miss. Laina	female	26.0	0	1	3101282	7.9250	S
1	1	Futrelle, Mrs. Jacques Heath	female	35.0	1	0	113803	53.1000	S
0	3	Allen, Mr. William Henry	male	35.0	0	1	373450	8.0500	S
0	3	Moran, Mr. James	male	35.0	0	1	330877	8.4583	Q

We consider 7 of the attributes as the input `features`: `Pclass, Sex, Age, Relatives, IsAlone, Fare, Embarked`. And consider the first attribute `Survived` as the `class label`. As shown in Table. 4, this is what you will read from the csv file.

Table 3: Sample data in the `titanic-train.data, titanic-train.label`.

Survived	Pclass	Sex	Age	Fare	Embarked	relatives	IsAlone
0	3	0	22	7	0	1	0
1	1	1	38	71	1	1	0
1	3	1	26	7	0	0	1
1	1	1	35	53	0	1	0
0	3	0	35	8	0	0	1
0	3	0	35	8	77	0	1

In order to read the CSV data and obtain our features and labels with the necessary attributes, you can use the following code:

```
import pandas as pd
X = pd.read_csv(data_file, delimiter=',',  index_col=None, engine='python')
```

Input format

Your python script should take the following arguments:

1. `trainFolder`: path to the training set folder. (`train-file.data, train-file.label`)
2. **testFolder**: path to the test set folder. These two files are **NOT** given to you. We will evaluate your result based on the Accuracy of your model on the testing set. Make sure your code can work with this argument. (`test-file.data, test-file.label`)
3. `model`: model that you want to use. In this case, we will use:
   - `vanilla`: the full decision tree.
   - `depth`: the decision tree with static depth.
   - `minSplit`: the decision tree with minimum samples to split on.
   - `postPrune`: the decision tree with post-pruning.
4. `crossValidK`: The number of cross validation steps.

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

```
import argparse

parser = argparse.ArgumentParser(description='CS373 Homework2 Decision Tree')
parser.add_argument('--trainFolder')
parser.add_argument('--testFolder')
parser.add_argument('--model')
parser.add_argument('--crossValidK', type=int, default=5)
args = parser.parse_args()
print(args)
```

Your code should read the training set from `trainFolder`, extract the required features, train your decision tree on the training set, and test it on the test set from `testFolder`.

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.

Entropy and Information Gain

Entropy Calculation

To help you build the model easier, we provide a code snippet for you to calculate the entropy:

```
import numpy as np
def entropy(freqs):
    """
    the equation is: entropy(p) = -SUM (Pi * log(Pi))
    >>> entropy([10.,10.])
    1.0
    >>> entropy([10.,0.])
    0
    >>> entropy([9.,3.])
    0.811278
    """
    all_freq = sum(freqs)
    entropy = 0
    for fq in freqs:                                    # you need to fill ______
        prob = ____ * 1.0 / ____
        if abs(prob) > 1e-8:                            # avoid computing log2(0)
            entropy += -____ * np.log2(____)
    return entropy
```

Here is the mathematical calculation steps of the above test cases:

$$-10/20 * \log(10/20) - 10/20 * \log(10/20) = 1$$
$$-10 * \log(10/10) - 0/10 * \log(0/10) = 0$$
$$-9/12 * \log(9/12) - 3/12 * \log(3/12) = 0.811278$$

make sure you can have the identical output when feeding the same input.

Information Gain

we provide sample code for you to calculate the information gain:

```
def infor_gain(before_split_freqs, after_split_freqs):
    """
    gain(D, A) = entropy(D) - SUM ( |Di| / |D| * entropy(Di) )
    >>> infor_gain([9,5], [[2,2],[4,2],[3,1]])
    0.02922
    """
    entropy_of_D = entropy(____)
    size_of_D = sum(____)
    weighted_sum = 0.
    for freq in after_split_freqs:
        Di_D_ratio = sum(____) * 1.0 / ____
        weighted_sum += ratio * entropy(___)
    gain_D_with_A = entropy_of_D - weighted_sum
    return gain
```

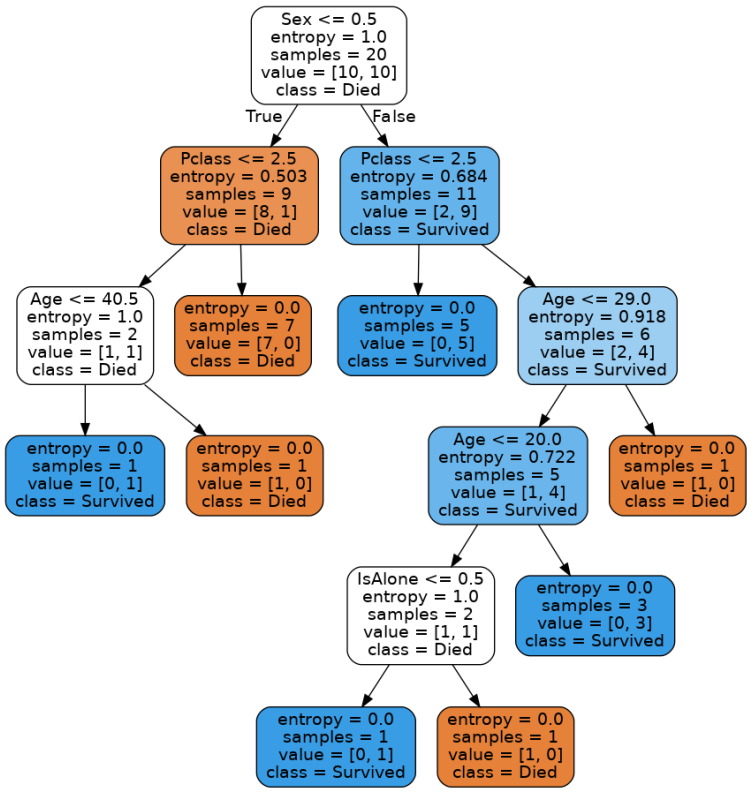
where you need to pass the test case:

$$entropy(D) = 9/14 * \log(9/14) - 5/14 * \log(5/14) = 0.9402$$
$$entropy(Income = high) = -2/4 * \log(2/4) - 2/4 * \log(2/4) = 1$$
$$entropy(Income = med) = -4/6 * \log(4/6) - 2/6 * \log(2/6) = 0.91829$$
$$entropy(Income = low) = -3/4 * \log(3/4) - 1/4 * \log(1/4) = 0.81127$$

$$
\begin{aligned}
Gain(D, Income) =& entropy(D) \\
& - (entropy(Income = high) + entropy(Income = med) + entropy(Income = low)) \\
=& 0.9402 - (4/14 * 1 + 6/14 * 0.91929 + 4/14 * 0.81128) = 0.02922
\end{aligned}
$$

Sample Decision Tree

Here is the example decision tree by using the `sample.data, sample.label`:



Here is the structure for helping you start this homework you can also find it in the python source file: `yourusername-ID3.py`.

```
class Node(object):
    def __init__(self, left, right, attribute, threshold):
        self.left_subtree = left
        self.right_subtree = right
        self.attribute = attribute
        self.threshold = threshold
```

One possible way to recursive way to recursively build the tree is to call a function `ID3`:

```
def ID3(train_data, train_labels):
    # 1. use a for loop to calculate the infor-gain of every attribute
    for att in train_data:
        # 1.1 pick a threshold
        # 1.2 split the data using the threshold
        # 1.3 calculate the infor_gain
    # 2. pick the attribute that achieve the maximum infor-gain
    # 3. build a node to hold the data;
    current_node = Node(None, None, the_chosen_attribute, the_chosen_threshold)
```

```
# 4. split the data into two parts.
# 5. call ID3() for the left parts of the data
left_subtree = ID3(left_part_train_data, left_part_train_label)
# 6. call ID3() for the right parts of the data.
right_subtree = ID3(right_part_train_data, right_part_train_label)
current_node.left_subtree = left_subtree
current_node.right_subtree = right_subtree
```

Part 1 Decision Trees (80pts)

Note: You need to finish only your code as a separate file (`yourusername-ID3.py`) for this section.
Your algorithm should finish training and testing within **5mins**. We will verify your training and testing modules, and you should not include your pre-trained model.

1. (**20 points**) Implement a binary decision tree with no pruning using the ID3 (Iterative Dichotomiser 3) algorithm[1]. Format of calling the function and accuracy you will get after training:

```
$ python3 yourusername-ID3.py  --trainFolder ./path/to/train-folder \
                  --testFolder ./path/to/test-folder \
                  --model vanilla \
                  --crossValidK 5
fold=1, train set accuracy= ____, validation set accuracy= _______
fold=2, train set accuracy= ____, validation set accuracy= _______
fold=3, train set accuracy= ____, validation set accuracy= _______
fold=4, train set accuracy= ____, validation set accuracy= _______
fold=5, train set accuracy= ____, validation set accuracy= _______
Test set accuracy= ____
```

   Make sure your program follows our suggested format. Make sure you have 5-fold cross validation and use the majority value for the missing value in the dataset.

2. (**20 points**) Implement a binary decision tree with a given maximum depth. The format of calling the function and accuracy you will get after training:

```
$ python3 yourusername-ID3.py --trainFolder ./path/to/train-folder \
                  --testFolder ./path/to/test-folder \
                  --model depth \
                  --crossValidK 5 \
                  --depth 10
fold=1, train set accuracy= ____, validation set accuracy= _______
fold=2, train set accuracy= ____, validation set accuracy= _______
fold=3, train set accuracy= ____, validation set accuracy= _______
fold=4, train set accuracy= ____, validation set accuracy= _______
fold=5, train set accuracy= ____, validation set accuracy= _______
Test set accuracy= ____
```

   Make sure your code has this extra argument `--depth`. The last argument (`14`) is the value of maximum depth.

3. (**20 points**) Implement a binary decision tree with a given minimum sample split size. Format of calling the function and accuracy you will get after training:

```
$ python3 yourusername-ID3.py --trainFolder ./path/to/train-folder \
                  --testFolder ./path/to/test-folder \
                  --model minSplit \
                  --crossValidK 5 \
                  --minSplit 2
fold=1, train set accuracy= ____, validation set accuracy= _______
fold=2, train set accuracy= ____, validation set accuracy= _______
fold=3, train set accuracy= ____, validation set accuracy= _______
fold=4, train set accuracy= ____, validation set accuracy= _______
fold=5, train set accuracy= ____, validation set accuracy= _______
Test set accuracy= ____
```

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

Make sure your code has this extra argument `--minSplit`. The last argument (2) is the value of minimum samples to split.

4. (**20 points**) Implement a binary decision tree with post-pruning using Reduced Error Pruning. Format of calling the function and accuracy you will get after training:

```
$ python3 yourusername-ID3.py --trainFolder ./path/to/train-folder \
                  --testFolder ./path/to/test-folder \
                  -- model postPrune \
                  --crossValidK 5
fold=1, train set accuracy= ____, validation set accuracy= _______
fold=2, train set accuracy= ____, validation set accuracy= _______
fold=3, train set accuracy= ____, validation set accuracy= _______
fold=4, train set accuracy= ____, validation set accuracy= _______
fold=5, train set accuracy= ____, validation set accuracy= _______
Test set accuracy= ____
```

FINAL CHECK

1. The tree building function and prune function should be implemented in a recursive manner, otherwise your solution score will be penalized by 50%.
2. The exact running time of your code should be within 5mins. If it takes more than 5mins, your solution score will be penalized by 50%.
3. Your program should works for argument "test-folder" and produce valid result. Otherwise, your solution score will be penalized by 50%.
4. Your program output format should follow our example (Check the Part 1 Decision Trees). Otherwise, your solution score will be penalized by 50%.
5. Your program by default should have 5-fold cross validation function and use the majority value for the missing value of the attributes.
6. your filename should be `yourusername-ID3.py`. If the instructor creates this file, it should be: `dan-ID3.py`.

Part 2 Analysis(20pts)

Note: You need to submit only your answers in the PDF for this section to Gradescope.
For the following questions, use `titanic-train.data, titanic-train.label` as the training file and `titanic-test.data, titanic-test.label` as the test file. You should use `numpy, matplotlib, seaborn` for plotting the graph, and include your code scripts. Make sure your `xaxis, yaxis, title` has proper name.

1. (**4 points**) Given the following table, compute the **Entropy** of the data and the **Information Gain** of choosing the "IsAlone" attribute. For the missing value, consider the following methods: 1) assign missing value with the most common one; 2) assign missing value with the median one; 3) throw the data sample away with any missing value.

Table 4: Sample data in the `titanic-train.data, titanic-train.label`.

Survived	Pclass	Sex	Age	Fare	Embarked	relatives	IsAlone
0	3	0	22	7	0	1	0
1	1	1	38	-	1	1	0
1	3	1	26	7	0	0	1
1	1	1	35	53	0	1	-
0	3	0	35	8	0	0	1
0	-	0	35	8	77	0	1

2. (**4 points**) Why don't we prune directly on the test set? Why do we use a separate validation set? Why cross validation can help reduce the over-fitting problem?

3. (**4 points**) For the decision tree (`vanilla`, In part 1 problem 1), measure the impact of different methods for the missing value (see part 2 problem 1) over the training accuracy and validation accuracy.

4. (**4 points**) Repeat the same analysis for the static-depth case (`depth`). Consider values of maximum depth from $\{2, 4, 6, 8, 10, 12, 14, 16\}$ and pick the depth with the best cross validation accuracy. Finally, plot the choice of depth against the number of nodes in the tree. Also plot the choice of depth against the the training set and validation set accuracy.

5. (**4 points**) Repeat the above analysis for the pruning case (`prune`). Consider values of minSplit from $\{2, 4, 6, 8, 10, 12, 14, 16\}$ and pick the minimum number of sample for split with the best cross validation accuracy. Finally, plot the choice of minSplit against the number of nodes in the tree. Also plot the choice of minSplit against the the training set and validation set accuracy.

Part 3 Bonus(5pts)

For this part, we will investigate the PCA algorithm for the decision tree. Use the dataset in the `bonus` folder and here is the code for you to load the data and the label

```
X = np.genfromtxt('bonus_data.csv', delimiter=",", dtype = float)
data = X[1:,:-1]
label = X[1:,-1]
```

1. you need to implement the PCA, with the following code structure

   ```
   class PCA(object):
       def __init__(self, n_component):
           self.n_component = n_component

       def fit_transform(self, train_data):
           #[TODO] Fit the model with train_data and
           # apply the dimensionality reduction on train_data.

       def transform(self, test_data):
           #[TODO] Apply dimensionality reduction to test_data.
   ```

2. first apply the PCA over the `data` and then apply the decision tree using the follow the command line:

   ```
   $ python3 yourusername-ID3.py  --trainFolder ./path/to/train-folder \
                     --testFolder ./path/to/test-folder \
                     --model vanilla \
                     --crossValidK 5 \
                     --PCANcomp 100
   ```

   We will verify your result over the testing set (this is not given to you). Make sure your code can work with the argument of **-PCANcomp**, which means the number of components for PCA algorithm.