HW1, Zhiping Xiu

Problem 1:

A):

No:

Two hypothesis function output same result, given D, doesn't mean they are same.

Two different hypothesis h1 and h2 can have same output, given D. In this situation, we can have two H_D, {h1 + rest of distinguishable hypothesis} and {h2 + rest of distinguishable hypothesis}, given D. Thus there are no unique H_D.

B):

H_D = |L| ^ |D|:

When we consider H_D, we assume H = C. The concept space size is calculated by |L| ^ |D|, So, the size of H_D is the same.

C):

h and c is distinguishable, given D, if they differ in their labeling of at least one of the examples in D.

Easy to know, h and c is indistinguishable, given D, if h and c is consistent ($\forall x \in D$, $c(x) = h(x)$).

The size of Concept space, which in this case should be our largest set of distinguishable hypotheses, should be H_D = |L| ^ | D| = 2 ^ 100 ~ 10 ^ 30. In the worst case, we will find our h in the last element of HD.

And for each element in HD, we will compare at worst |D| (but normally |D|/2) equation to see if h(x) = c(x) for $\forall x \in D$.

Let's assume consider 10^9 (one billion) hypotheses per second means we don't care about how many (if h(x) == c(x)) is calculated, the time spent would be at worst 2^100 / 10^9 ~ 10^30 / 10^9 ~ 10^21 seconds. This is not a reasonable time to wait.

D):

The size of concept space for |X| is 2^200 ~ 10^60.

The size of concept space for |D| is 2^100 ~ 10^30.

If a hypothesis h indistinguishable with target concept, given | X|, h will also indistinguishable with target concept, given |D|, NOT vice versa.

So, the answer would be:
Pr(h | ∀x ∈ D, h(x) = c(x)) / Pr(h | ∀x ∈ X, h(x) = c(x))
= (1 / 10^60) / (1 / 10^30)
= (1 / 10^30)

(The notation might be inappropriate, because I miss some probability background, but the Pr(h | ∀x ∈ D, h(x) = c(x))'s idea is the probability of selecting one hypothesis h from hypothesis space, given D, that this hypothesis meet the condition that h is distinguishable with target concept c, given D. BTW, if the notation is indeed inappropriate, can you help me correct that? thanks!)

Problem 2:

A):

Node1

choose root node from:

[country, manufacturer, color, decade, type]
country: gain([3, 2], [[3,1], [0, 1], [0, 0], [0, 0]]) = 0.321928
manufacturer: gain([3, 2], [[2, 0], [1, 1], [0, 1]]) = 0.57095
color = gain([3, 2], [[2, 0], [0, 1], [0, 1], [1, 0]]) = 0.97095
decade = gain([3, 2], [[0, 1], [2, 1], [1, 0], [0, 0]]) = 0.419973
type = gain([3, 2], [[3, 1], [0, 1], [0, 0]]) = 0.321928

(The gain is calculated by my code, please see how gain() gain work in my code p5_code/helper.py)

We choose the most gain branch, which is Color

No need to consider next node, recursive ended because all sub node are fully divided.

The tree would be like:

```
—x.color?
——Blue:     True
——Green:    False
——Red:      False
——White:    True
```

B):

Encoded function: H(x) = x.color_is?(Blue) || x.color_is?(White)

# x.color_is?(param) will return True if x.color == param, return False otherwise.

It didn't catch the concept of "Japanese Economy Car" at all.

Why and what should we do?

1: The data set is too small, and just happened that one irrelevant attribute get the most entropy gain.

2: We should introduce some "bias". Since the concept is Japanese Economy car, we should focus on attribute like "Origin" and "Type". Because this attributes are more likely to catch the meaning of the concept.

Problem 3:

A):

One naive solution I think is to use the mean value of training data as split point. Say in training data we have [1.,2.,3.,4.,5.,6.], we will use 3.5 as split point, less than 3.5 go left, go right otherwise.

Good thing about it is it's easy to implement, and do not take too much time to make the decision. And it can output a reasonable answer for certain dataset.

Weaknesses about it is, it won't able to consider more complex situation, like what if we want to divided it to three branch (we can, however, use 2*mean*1/3 and 2*mean*2/3 as split point, but its might be not so good). Or what if the dataset is looking like the first graph in question 3-B.

B):

The second graph can be easily represented, we can choose .5 as split point for attributes in both dimension.
However, considering ID3 were not able to consider two attribute in the same node, and the fact that we are only able to select one optimal split point, I think maybe there are no good way to represent the first graph. If we are allowed to select multiple split

point, we may try to use tons of split point to approximate the concept, but we are not allowed to do so, sadly.

Problem 4:

A):

True Positive(TP): what our AI think is true, and luckily, it is true.
True negative(TN): what our AI think is false, and luckily, it is indeed false.
False Positive(FP): what our AI think is true, but sadly, our AI is wrong, it is actually should be wrong.
False negative(FN): what our AI think should be false, but in fact it's true.

Basically, Positive or negative are what our AI think the data should be predicted. True or False is if our AI is correct, or not

Source: https://en.wikipedia.org/wiki/Precision_and_recall

B):

Precision: TP / TP + FP, indicating how can we trust our AI's prediction.
Recall: TP / TP + FP, indicating how good our AI's can cover as many true positive answer in the training data set.

Source: https://en.wikipedia.org/wiki/Precision_and_recall

C):

F1 score is the harmonic mean of precision and recall.
Or F1 = (2 * precision * recall) / (precision * recall).

Source: https://en.wikipedia.org/wiki/Precision_and_recall

I made up a toy example about 4-B) and 4-C) before, see more in http://www.zephyrsails.com/articles/10, if you like.

D):

Confusion matrix is a matrix built by considering: how our AI predict, and what the data actually is.

E.g. Say our AI is about to look at bunch of pictures and tell us which one is bird, which is dog, which is cat. The confusion matrix should be:

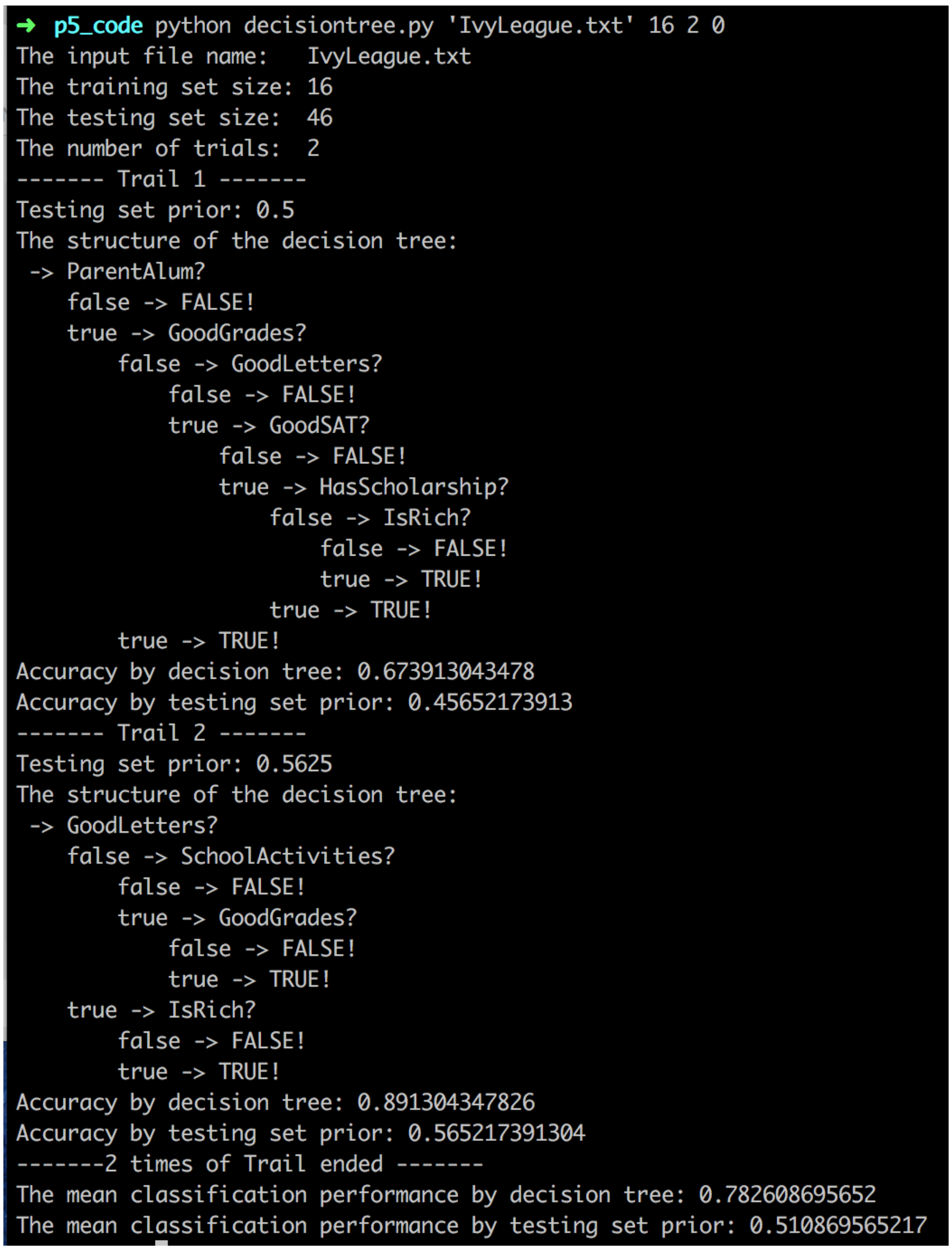
	Predicted Bird	**Predicted Cat**	**Predicted Dog**	
Actual Bird	TB=80	FC=10	FD=10	100
Actual Cat	FB=10	TC=80	FD=10	100
Actual Dog	FB=10	FC=10	TD=80	100
	100	100	100	300

By using this matrix it can kind of tell us how smart our machine learner is, and see how it may make mistake. E.g. We can see the Accuracy is 80%, which means it predict correct answer in 80% input.

Source:
https://en.wikipedia.org/wiki/Precision_and_recall
http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

Problem 5:

Please see /p5_code，The output should be like this:
If you have any trouble with this code, please send a screenshot to zhipingxiu2018@u.northwestern.edu instead of just grade me zero, please.



```
➜  p5_code python decisiontree.py 'IvyLeague.txt' 16 2 0
The input file name:   IvyLeague.txt
The training set size: 16
The testing set size:  46
The number of trials:  2
------- Trail 1 -------
Testing set prior: 0.5
The structure of the decision tree:
 -> ParentAlum?
    false -> FALSE!
    true -> GoodGrades?
        false -> GoodLetters?
            false -> FALSE!
            true -> GoodSAT?
                false -> FALSE!
                true -> HasScholarship?
                    false -> IsRich?
                        false -> FALSE!
                        true -> TRUE!
                    true -> TRUE!
        true -> TRUE!
Accuracy by decision tree: 0.673913043478
Accuracy by testing set prior: 0.45652173913
------- Trail 2 -------
Testing set prior: 0.5625
The structure of the decision tree:
 -> GoodLetters?
    false -> SchoolActivities?
        false -> FALSE!
        true -> GoodGrades?
            false -> FALSE!
            true -> TRUE!
    true -> IsRich?
        false -> FALSE!
        true -> TRUE!
Accuracy by decision tree: 0.891304347826
Accuracy by testing set prior: 0.565217391304
-------2 times of Trail ended -------
The mean classification performance by decision tree: 0.782608695652
The mean classification performance by testing set prior: 0.510869565217
```