## CS365 Lab C Report Huong Phan

### I. Design

- 1. Read the data files and store examples
- First, import pandas, math package
- Create a function called read\_file that takes in a file name as input, and returns a data frame, a few lines of which look like this if we consider the tennis.txt file:

	outlook	temperature	humidity	wind	playtennis
0	sunny	hot	high	weak	no
1	sunny	hot	high	strong	no
2	overcast	hot	high	weak	yes
3	rain	mild	high	weak	yes
4	rain	cool	normal	weak	yes

- 2. Calculate information gain for each attribute:
- Create a function called entropy that takes a set of examples and calculates the entropy for the whole set.
- + Equation: Let  $P(V = v_k) = \frac{\# \ of \ observations \ for \ a \ certain \ value}{total \ \# \ of \ observations}$ , V is a random Boolean variable with values  $v_k$ . Since a Boolean random variable only has a positive or negative value, the entropy for that variable is as follows:

$$B(q) = -[P(q)\log_2 P(q) + (1-q)\log_2 (1-q)]$$

in which q to be the possibility of positive examples over all examples =  $\frac{p}{p+n}$ , p is the number of positive examples, n is number of negative examples. The entropy of the goal attribute of whole set is:

$$H(Goal) = -\left[\frac{p}{p+n}\log_2\frac{p}{p+n} + \frac{n}{p+n}\log_2\frac{n}{p+n}\right] = B(\frac{p}{p+n})$$

- Create a function called importance that takes a set of examples and an attribute and calculates the information gain from splitting on that attribute
- + For a single attribute A with d distinct values, if we split on that attribute, we would split the whole set into d subsets with a distinct value on attribute A, with  $p_k$  and  $n_k$  being the number of positive and negative examples in the  $k^{th}$  subset. We can calculate the expected entropy remaining after testing A as follows:

Remainder(A) = 
$$\sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B(\frac{p_k}{p_k + n_k})$$

in which  $B(\frac{p_k}{p_k+n_k})$  is the entropy of Boolean variable over a single subset with a distinct value of attribute A, while  $\frac{p_k+n_k}{p+n}$  is possibility of randomly choosing an example of training set that belongs to  $k^{th}$  subset.

- Create a function called importance that takes a set of examples and an attribute A and calculates the information gain from splitting on that attribute:
- + Information gain = H(Goal) Remainder(A)

### 3. Storing our tree:

- Leaf node has the following attribute:
- + classification: string, initialized to be None (e.g. 'yes', 'no')
- Tree node would have the following characteristics:
- + attribute: string (e.g. humidity)
- + label: string, initialized to be None, format: <attribute = value>

(e.g. 'outlook = sunny')

- + subtree: leaf object, initialized to be None
- 4. Build the tree recursively using Decision-Tree-Learning Algorithm:
- Create a function called plurality\_value, that takes in examples as input and return the most common classification of the examples. If a tie happens, return 'no'.
- Create a function build\_tree that takes in set of examples, attributes, parent examples as inputs and return a tree and print the tree into a text file. The algorithm is based on the Decision-Tree-Learning Algorithm in the textbook, p. 702.
- Note: The tree implementation has some problems, so there is some problem
  with accessing components of the tree, like if I call tree.label it only gives me the
  last label of the tree. I print the tree into a text file and use that file for any
  classification that happens later on.

### 5. Display the tree:

- Create a function called display\_tree which takes a filename as input and return the visual representation of the decision tree in a text file and number of nodes in the tree.
- 6. Training set accuracy:
- Create a function called training that takes the set of examples and the index of one example as inputs.

- + Build a tree from the set of all examples and, then use the decision tree to predict the classification for each example.
- + Return True if the prediction matches with the real classification, False otherwise.
- Create a function called training\_set\_accuracy that takes a filename as input, return the training set accuracy percentage.
- + Run for n times, n = total number of examples, accuracy = number of True's/n

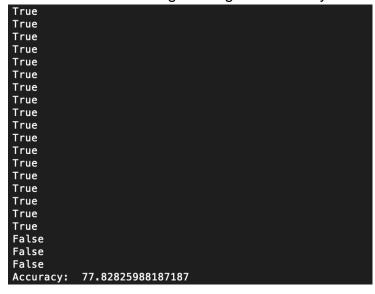
### 7. Accuracy testing:

- Create a function called leave\_one\_out\_cross\_validation that takes the set of examples and the index of one example as inputs
- + Build a tree from the set of all examples except for the one example with the index above, then use the decision tree to predict the classification for the one example that was left out.
- + If an example only appears uniquely once in the set, predict it as 'no'
- + Return True if the prediction matches with the real classification, False otherwise.
- Create a function called accuracy\_testing that takes a filename as input, return the accuracy percentage of cross-validation.
- + Run for n times, n = total number of examples, accuracy = number of True's/n

#### II. Results

	tennis	pets	titanic2
number of nodes in the tree	12	15	14
training set accuracy	100%	73.3%	77.8%
testing set accuracy	92.9%	46.6%	77.8%

A screenshot of running training set accuracy for titanic2.txt



# Decision trees for:

tennis.txt	titanic2.txt	pets.txt
outlook = sunny	sex = male	size = tiny
humidity = high	pclass = 1st	color = white
		toil – vos
no Book to humiditur	age = adult	tail = yes
Back to humidity:		 
h	no Book to once	earshape = pointed
humidity = normal	Back to age:	
IIII		no Baal ta aasal aasa
yes	age = child	Back to earshape:
Back to humidity:	IIIII	 D. 14 4 7
	yes	Back to tail:
Back to outlook:	Back to age:	
		Back to color:
outlook = overcast	Back to pclass:	
IIIII		color = brown
yes	pclass = 2nd	IIIII
Back to outlook:		no
	no	Back to color:
outlook = rain	Back to pclass:	
IIIII		Back to size:
wind = weak	pclass = 3rd	
IIIII		size = small
yes	no	IIIII
Back to wind:	Back to pclass:	yes
		Back to size:
wind = strong	pclass = crew	
IIIII		size = medium
no	no	
Back to wind:	Back to pclass:	no
		Back to size:
Back to outlook:	Back to sex:	
		size = large
	sex = female	
	IIIII	no
	yes	Back to size:
	Back to sex:	
		size = enormous
		IIIII
		no
		Back to size:
		·····