Decision tree

Working with the data

- Accept as a matrix
- Add another column that holds which subset it is a part of
- Implement the tree
- function at the end classify which and takes data returns the classes

1. Initial Decision Tree

- Write a function *learnDecisionTree* that takes as paremeters
  - A matrix of variable values (data set)
  - vector of classifications
  - *e.g.* The meas matrix and specisies vector produced by loading the Fisher Iris data set
  - Attach column for classification
  - make up a split and what variable - what number
  - split the variable based on those parameters
- Have the data together
- The function *learnDecisionTree* should construct a data structure appropriate for representing an increasing number of data sets
- The data sets will be produced by repeatedly splitting the intial data set according to the decision tree learning algorith presented in class
- This function should also call the functions below
  - **Intially to test them**
- Eventually to implement the full decision tree learning algorithm
- inputs: M (matrix), v(classification vector)
- process: Splits input via the classification repeatedly
- outputs: Constructs a data structure appriopriate for representing data

Entropy

Measure of impurity It is defined for a binary class with values a / b as: - Entropy = - p(a) * log(p(a)) - p(b) * log(p(b)) - It reaches its maxinum when the probability is p = 1/2 - Meaning that p(X=1)=0.5 or similarly p(X=b)=0.5 - Having a 50% / 50% chance of being either a or b - **Uncertainty is at a maximum** - Entropy function is at zero minimum when probability is p=1 or p=0 - with complete certainty

Need it to calculate information gain

Using this example:

```
     ends-vowel
      [9m,5f]          <--- the [..,..] notation represents the class
    /          \            distribution of instances that reached a node
   =1           =0
 -------     -------
 [3m,4f]     [6m,1f]
```

Entropy before Steps: 1. Find the difference of the log function for each class before split entropy before = (5/14) * log^2(5/14) - (9/14) * log^2(9/14) = 0.9403 ^ female/total ^ male/total

Entropy left Steps: 1. Find the difference of the log function left of the split entropy left = (3/7) * log^2(3/7) - (4/7) * log^2(4/7) = 0.9852 ^ female/total(after split) ^ male/total(after split)

Entropy right Steps: 1. Find the difference of the log function right of the split entropy right = (6/7) * log2(6/7) - (1/7) * log2(1/7) = 0.5917

Entropy after Steps: 1. We combine the left/right entropies using the number of instances down each branch as weight factor - 7 instances whent left, and 7 instances went right - The final entropy after the split: entropy after = 7/14 * Entropy left + 7/14 * Entropy right = 0.7885

Infromation gain This measure or purity is called the information - It represents the expected amount of information that would be needed to specify wheather a new instance should be classified based on the rules. Information gain = entropy before - entropy after = 0.1518

Interpation of the above calculation - By doing the split with the `end_vowels` feature, we were able to reduce uncertainty in the sub-tree prediction outcome by a small ammount of 0.1518

Your Task

- Load a data set and augment with set index and, max gain
- A function for calculating the entropy of a data set
  - entropy(S)
- A function for splitting a data set given a rule
  - [S1,S2] = split(S, varIdx, threshold)
- A function for calculating the information gain of a split set
  - gain(S,S1,S2)
- A function for generating rules
  - [varIdxs, thresholds] = candidates(S)
- A function for splitting sub-sets until all gains are 0 or negative