## [CAP4611-21Spring](https://webcourses.ucf.edu/courses/1369384/calendar_events/2158980)

# Day 6 (Tuesday, Feb 2):

(?) = missing details

**Quiz next Tuesday**

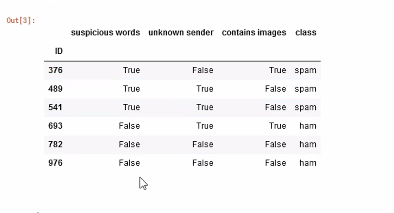
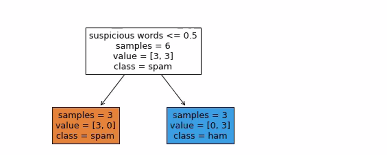
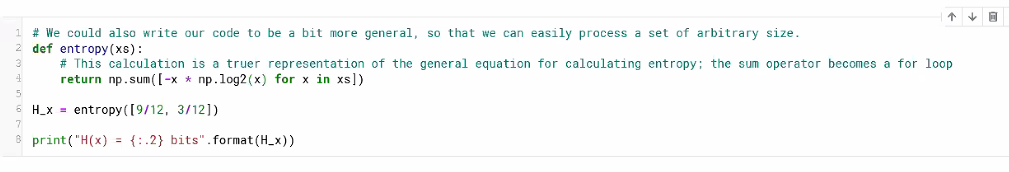
Assignments for the course:

* First assignment should be posted next week
* We will create a notebook with our code, generate an output file, submit it to a corresponding competition
* To get you ready, he created a practice assignment on kaggle
* There are a few tutorials out there on how to use kaggle
* Basically, your output will be compared to a list of known solutions
* “My suggestion is to go through the titanic tutorial for kaggle”
* The practice assignment is not graded
* “I suggest that you don’t necessary public to the assignment, as someone might copy you”
* For your assignments, don’t make your notebooks public… don’t associate them with the competition and make them public
* If you have questions, ask the discord or on one of the other comm channels
* So, when you actually submit your predictions, your results will be displayed on the leaderboard. The public leaderboard is ranked against of subsets of the solution file.
* You create your model, you submit your predictions, and you will see a score. If it is low, then go back to your model and regenerate your submission and check the score again
* You should have unlimited amount of submissions for the competitions (except the practice one?)
* You will not be graded relative to each other, if we did, it will not be great for our grades.
* “I will create a competition with a dataset, you download the dataset, you create a model and submit your predictions”
* **The first assignment is restricted to decision trees or reinforcement (what we have covered so far)**
* Private leaderboard, what is used to determine if someone “wins
* In case of the practice model, how I got a score of 37 on the competition,
  + I read in the test data
  + I assigned everything a zero
  + And I submitted it
* You can find the practice assignment in the announcements on webcourses
* What is the diff between public and private leaderboard?
  + The private leaderboards is a specific dataset to generate your models(?)
  + The public leaderboards uses a different dataset to test your models(?)
* Not all your grade is based on leaderboard score, but it is graded upon a benchmark
* The other half of the grade is based on your notebooks
* **For Kaggle, you don’t have to use your knights email if you don’t want**
* **For the first assignment, it will be done solo to stress test TAs**
* The size of the group reflects the difficulty or a problem. If you want large groups, you’ll have difficult problems

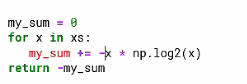
…

“Lets take a look at what we did last week”

Decision trees:

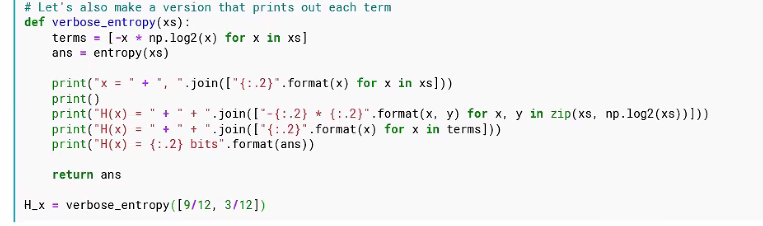
* [Recap] A decision tree allows us to classify data, make decisions
* Decision trees are (usually) binary trees
* It consists of a root node, leaf nodes, inner nodes(?). Each node is associated with a question
* Remember the spam?
  + Our goal in the spam problem is to decide whether an email is spam or not
  + Looking at the dataset
  + 
  + Is there a way to split the data up and create a decision tree?
  + Yes, we can create a decision tree to determine if the email is spam or ham
* Here is a decision tree:
  + 
* <= 5, if 0 go left, if 1 go left (T/F)
* Samples - amount of things in each resulting set
* Class is (?)
* The way we figured out what question to use is based on calculating entropy
* Remember that we use a base 2 logarithmic function
  + We use that because the equation is from information theory
* **High entropy has a low percentage of occurring**
* **Low entropy has a high percentage of occurring**
* Entropy equation:
* 
* So one way we can rewrite the equation, is with this line here (line 4):
* 
* xs is an element of xs,
* We are assuming that xs is some sort of sequence of the values (a python list)

Why are we doing it this way and not like this?



* List comprehension (line 4) of 2nd picture above is a lot faster, it is a standard in python. This has something to do with C being executed in the background

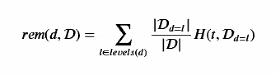
Verbose entropy:

* Basically showing your work
* 

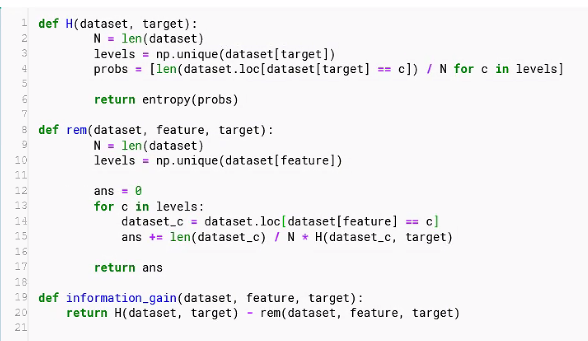
Profilers:

* “A profiler is your friend”
* A profiler is code that executes in the background and allows you to gather performance data on the code.
* The general approach is
  + You write your code
  + Decompose the code into small functions
  + Once you have your code written, run it with the profiler, gives you a report, tells you how long they take,
  + Find the functions that take a long time, and see if you can speedem up
* If you have a 5x nested for loop and it is running fast, then you don’t need to fix that
* **“Learn how to use a profiler”**
* “Having the knowledge of algorithms is great, but if your search routine is called only once in your entire program’s lifetime and its only sorting a small amount of data, it does not matter if you use bubble sort. You should account for the context that your code is written.”
* “If you run your code and you see that the bubble sort is running too slow, you might wanna turn it into quick sort”

We stopped at information gain last week:

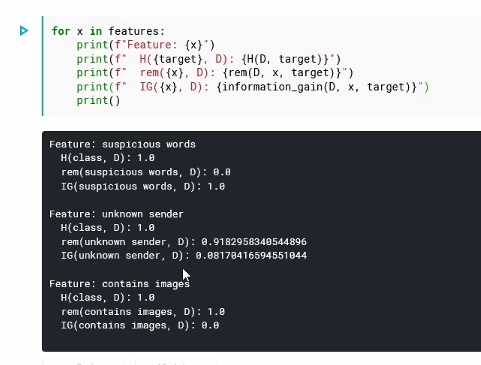
* Information gain is ONE of the ways you can figure out which feature to build your tree from
* The basic equation for information gain is:
* 
* The beginning amount of entropy - the remaining amount of entropy after splitting = information gained
* The middle equation:
* 
* You want the entropy of the subset of the data, and you want to scale it by the proportion of data in which that occurs. The vertical bars are the **Cardinality (**Total amount of elements**)**
* So what the information gain tells us is that we take the entropy of the target vector, take away some entropy from a subset, and whatever entropy remains is (?)

Writing code to do these calculations:

* 
* To calculate the entropy of the target vector given a dataset,
* You get the length of that dataset,
* you find how many levels
* You calculate the entropy
* For the remainder, you calculate size of dataset
* Calculate what are all the levels within the feature
* For every value in that level, you calculate a subset, and the entropy of the subset,
* (List comprehension isn’t done here because it is too complex)

**Recording starts**

* If we wanted to calculate the information gain for every one of these features in the dataset, we can just loop through all the features

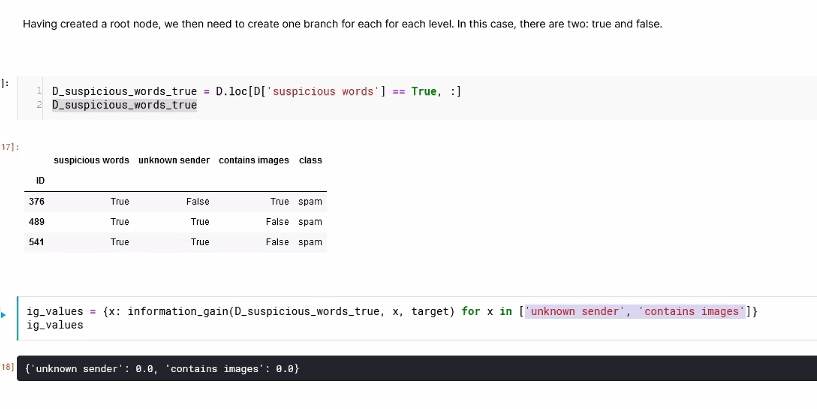


* We want to split on the one that has the most information gain.
* So this notion of information gain and entropy is what is called the ID3 algorithm

ID3 Algorithm:

* One of the OG decision algorithms, there are other ones out there though
* The ID3 Algorithm is straight forward
* **A greedy algorithm, trying to make the smallest tree possible**
* **Recursive algorithm**
* Find the best feature to split with
* Take the data split it,
* Repeat

Example:

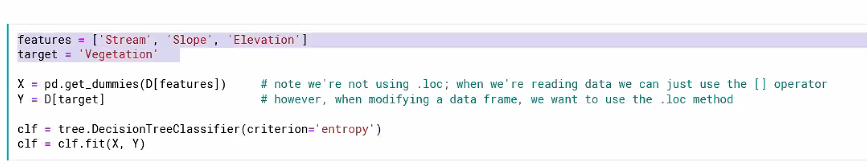
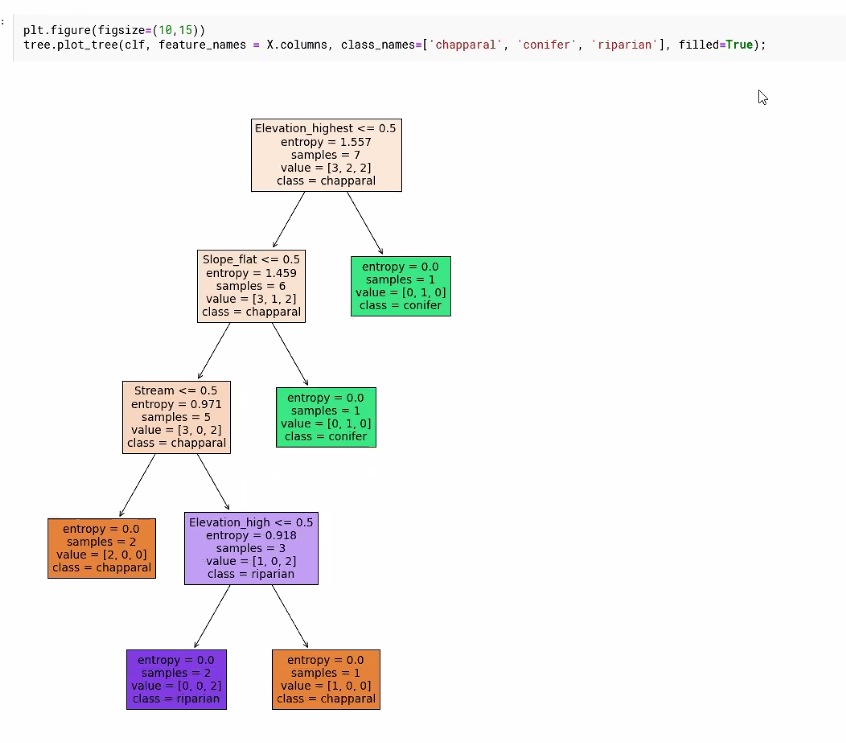


* Take the dataset, create two subsets
* On the two subset datasets, we calculate the information gained
* Since we split with sus words, we don’t include that in the resultant columns (?)
* If you want to automate this process, that is the purpose of the ID3 algorithm
* [Tip] the describe function will show us the count, unique, frequency, and top values
* 

At the end of the day, we don’t have to write a decision tree from scratch. We can use scikit or some other libraries

* Scikit-learn uses a (?) tree
* The problem with Scikit’s decision tree is
  + It does not know how to deal with categorical data
* So the way that you handle categorical data in most ML algorithms
  + You create a ONE-HOT encoding
    - You take your categorical feature, and split it up into multiple binary features
    - Pd.get\_dummies(D[[Column1, column2, column3 ...]])
  + We have to do this because this will result in something scikit can interpret
  + Encodings can also be done:
    - Using n-1 columns (?)
      * (?)
    - Set 0,0 to a default value
      * This will allow you to use two vectors as data (2^2 combinations (?))
  + Standard practice (as it works out nicely mathematically)
    - 1-hot encoding
    - N-1 encoding

Basic machine learning algorithm

* 
* You create your description features
* You create your target features
* You create your decision tree classifier
* You then run clf.fit(X,Y) builds the decision tree.
* Then plot the data
* 

For the assignments:

* Use the APIs, play around with the parameters
* Look at the documentation
* Decision trees are a way of codifying your data
* Another way we can calculate the entropy (??) is the guinian index
* It will have basically the same result

“Its not about the data you have, but the data you don’t have” - Dr. H

* How large does the dataset have to be?
  + All depends, but more is usually better

Out of time:

Thursday:

* Talking about extensions

Also:

* Textbook slides in the announcements, the author of the book has published slides for free to accompany the book
* “I am sticking close to what the slides cover, so you have a general idea of what we’re talking about in the lecture’
* “I am following what is listed on the schedule, which corresponds to the chapters in the book, which corresponds to the slide”
* The current stuff is most of the stuff in info based learning

**Kaggle notebook for today will be published**

...

End of recording