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

# Day 7 (Thursday, Feb 4):

(?) = missing details to be filled in

**Quiz on Chapter 4 next Tuesday**

**Quiz after that will be on Chapter 9**

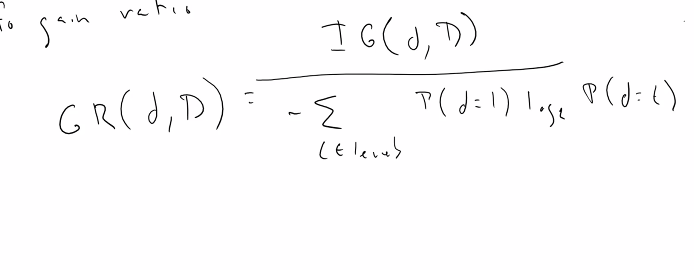
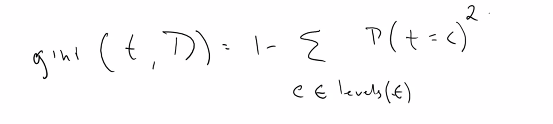
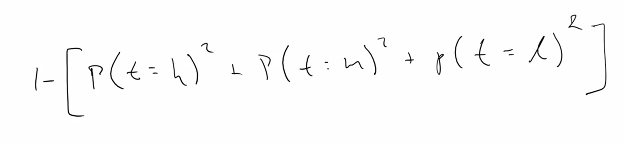
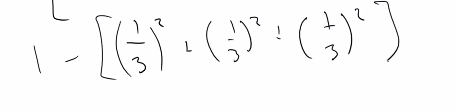
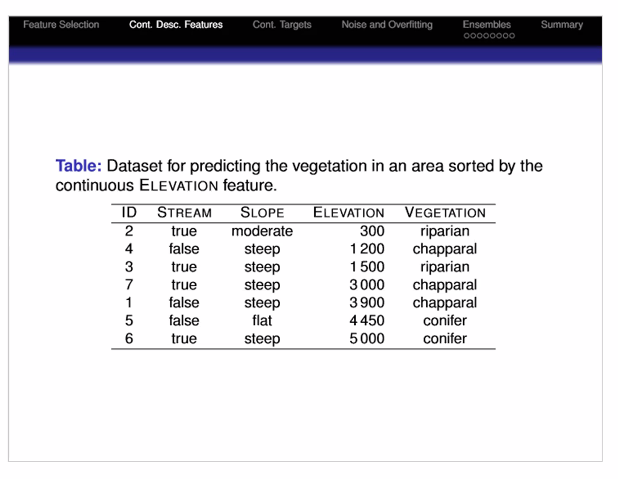
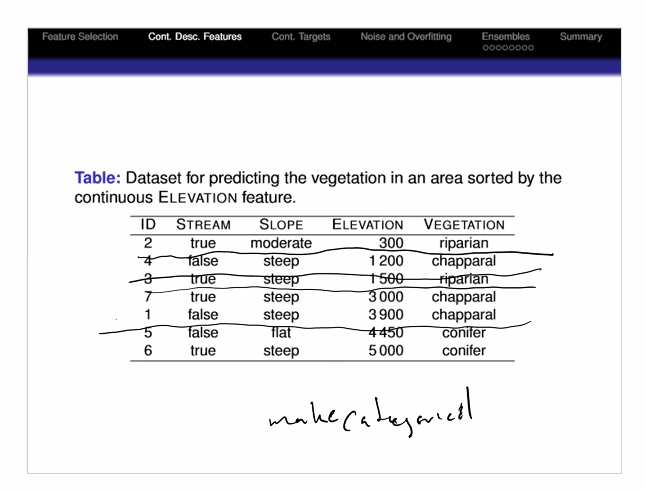
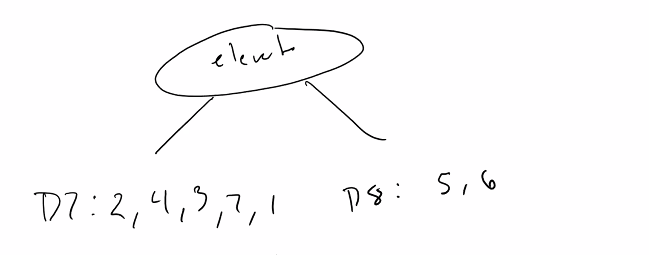
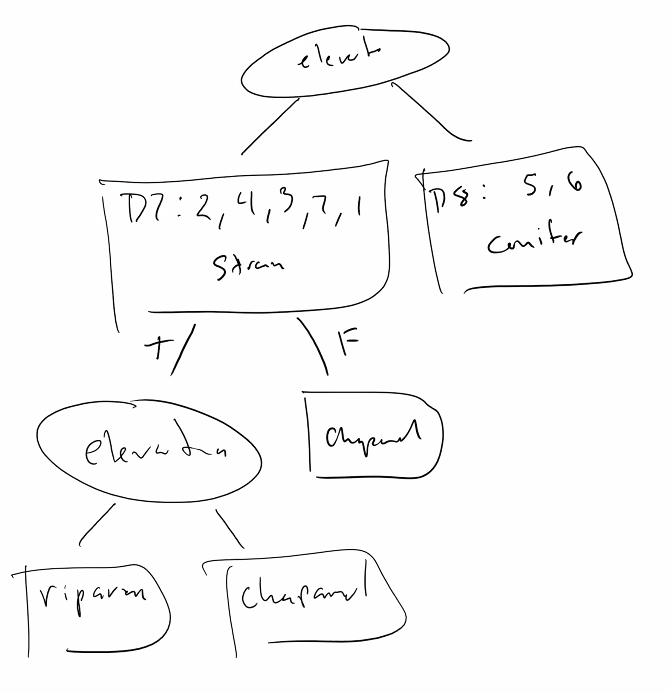
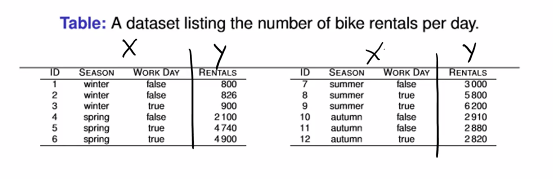
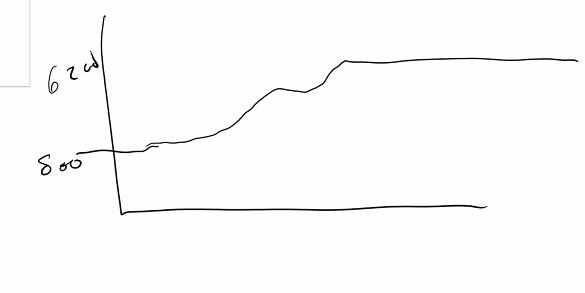
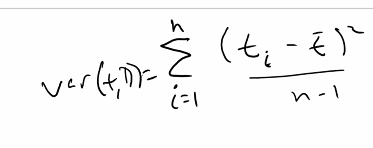
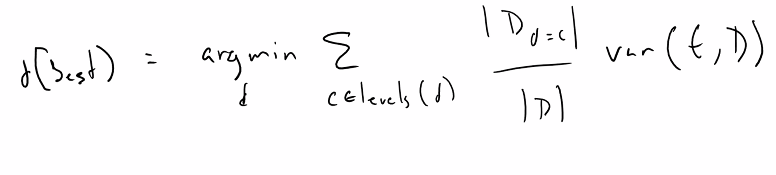
Webcourses:

* Reminder of online resources
* All questions pertain to the textbook
* The notes and slides are related to the book

….

Recording starts: 4:57

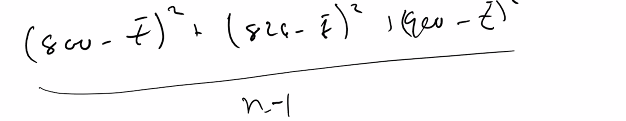
Recap:

* Up to this point we’ve talked about
  + Decision trees
  + Entropy
  + Information gain
  + Id3
* We have talked about how we can predict a target value based on an existing dataset by building a model, more specifically, a decision tree
* If you remember the bias at hte beginning of the week:
  + Our Bias will be just looking at trees
* Besides Entropy, we can measure information gain by:
  + Information Gain Ratio
  + Gini index
* Information gain GR(d,D) = informationGain(d, D) / the entropy of the (?) of the subdata
* We aren’t going to go too deep into this
* Normalizing is when you scale the value
* Entropy is used to calculate the information gain, but the information gain ratio is a measurement of the ratio of the subsets (?)
* When we calculate information gain, we can also calculate information gain ratio
* We can also use the **gini index**
* The gini index is defined as:
  + gini(t, D) = 1 - summation of levels of the probability of (t = 1)
  + 
  + There is also the gini coefficient and a few other variations
* Example
  + For the sake of argument, lets say:
  + T = [high medium low] = [h m L]
  + 
  + What is the probability that t=h? ⅓
  + So,
  + 
  + 1 - ⅓ = ⅔ bits
  + What if our vector represents [h h m]?
  + 1 - [⅔ ^2 + 1/9 + 0 ] = 1 - [4/9 + 1/9] = 4/9
  + 4/9 < ⅔
  + The gini coefficient is useful because it tells you the probability of usefulness thus it allows you to determine the information gain and pick the optimal question to ask
  + The reason that we have the square term in the gini coefficient is to make things very small. This allows us to differentiate between large and small information
  + [Note] With APIs, not everything you can do with a specific algorithm may not be supported by that library. Is the library good enough? Or do you have to make your own solution? scikit -learn does not support categorical features, you must use ONE-hot encoding or the other techniques we’ve discussed
* Look at this dataset
* 
* There is a continuous feature in this set, Elevation is a continuous value
* We will turn Elevation into a categorical variable, by selecting ranges of values
* **The way we turn quantitative information into categorical information is to**
  + Sort dataset based on the categorical data
  + Find out where the features differ,
  + Wherever the target features differ, that is where you want to put a break
  + 
  + We create bins based on these borders
    - To calculate the border for 300 and 1200, we take the mean which is <= 750
    - To calculate the border for 1200 and 1500, we take the mean which is <= 1350
    - To calculate the border for 1500 and 3000, we take the mean which is <= 2250
    - To calculate the border for 3900 and 4450, we take the mean which is <= 4175
  + Let us say we want to break up the data between row 2 and all the other rows.
    - We split the data with <=750, the information gain is 0.03060
    - If we split on 1350, the information gain is 0.1839
    - If we split on 2250, we get the value .5917
    - If we split on 4175, we get the value .8631
  + [Note] there is nothing that says that a decision tree needs to be balanced
  + So just like categorical features, the information gain tell us what to split on
  + This particular example:
  + 
  + Whenever we decide to split on elevation, this is one of the options
  + If we split on 4175, everything on the right tree is a conifer and no conifers appear in the left set. This is something we want.
  + Splitting at 4175 gives us the most information gain
  + Now, we’re dealing with another subset on the left, in our case, we are going to split on Stream [assuming we did a bunch of math beforehand]
  + 
  + We can then split again on elevation to differentiate between the two other categories
  + You can conceivable try splitting different ways. We made the boundaries using mean, but you could try a different way
* What if we have a dataset that looks like this?
* 
* What do we notice about this dataset?
  + The target is continuous
* Who wants to take a guess at what will happen if we tried to build a decision tree with the knowledge that we currently have?
  + It will end badly, too many leafnodes
* One thing that we can certainly try and do is that we can take a range approach. What if we did a bunch of ranges (binning?)
  + Our range of the target vector is [800, 6200]
  + But the problem is:
    - What if we have values outside of the training dataset that are outside of the range of these buckets?
  + You can do this, the buckets can be:
  + [0, 800] , [800, 1800], [1801, 2801], [5801, 6800], [>= 6801]
  + Is this actually useful?
    - Not really
    - If you plot this, you’ll get something that looks like
  + 
    - This is something you probably don’t want
    - [recall] there are two kinds of problems in machine learning:
      * Regression
      * Classification
    - For a **regression tree, which is used for continuous values**
      * In the case of a regression tree, we use something called variance
      * The equation we use to calculate information gain for a regression tree is variance.
      * 
      * Lets say we have the variance, and we have a node to split on
      * **The best feature is the one with the least variance**
* 
  + - * This equation says that the best feature to split on is the one with the least amount of variance.
      * If we were to look at the data, season and workday:
        + Season - how many different factors?

4, winter, fall, spring, summer

So if we split on winter, then we see that we have three values inside of winter. Elements 1, 2 ,3

The component that talks about size is 3 / 12 = .25

* + - 

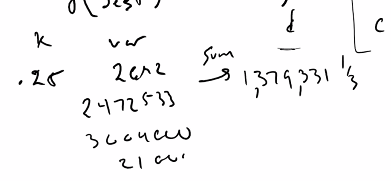
We then find out that our variance is 2.692 for .25

[note] Argmin means that we want defined some value for d that minimizes the equation.

Argmin of d tries to minimize the equation in terms of d

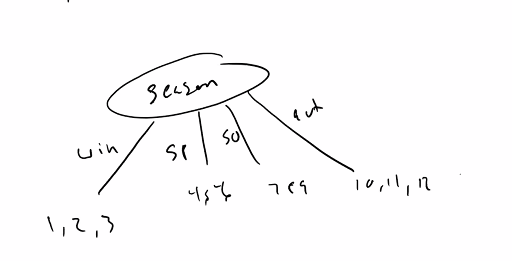
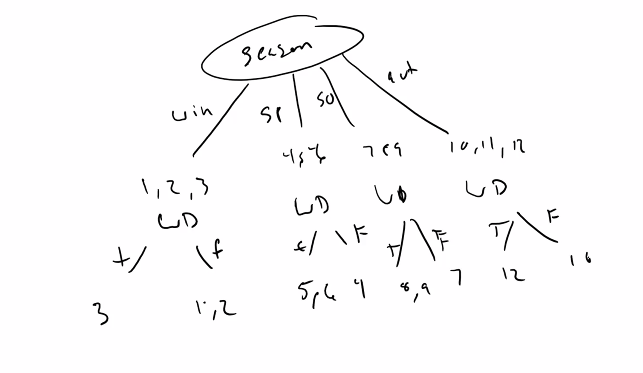
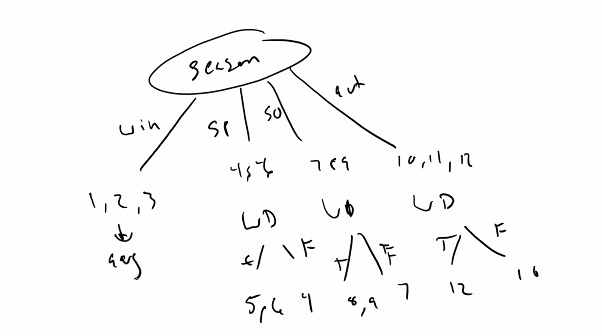
If we calculate the variance for the rest of them,

We get:

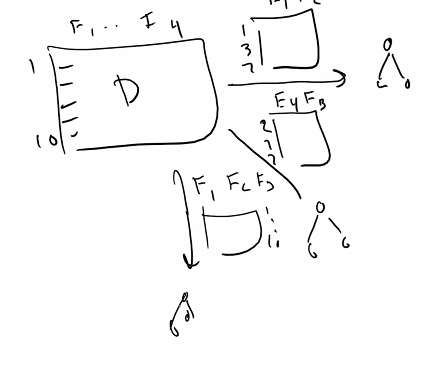


We then calculate the same values for work day which is 2,851,813 (?)

Therefore the smallest variance is within season. Thus we split based on season:

* + - * + 
        + For each of those datasets now is only one feature left.
* Since there is only one feature left, we can either choose to split it on work day, we end up with the dataset of:
* 
* The only thing left now is the target vector
* Now we choose what to return on
* You may choose to return based on the average, median, or max (for some reason)
* But basically, you aggregate the leaf nodes and return that data
* Note that when the tree gets too big:
  + It overfits
* One of the ways you can remove overfitting is that you can **prune the tree**, cut some of the branches from the tree and return the average [or maybe some other aggregate value]
* 
* Post pruning is a more common approach:
  + You build the entire tree
  + And remove parts of the tree to see how it changes the error (?)
  + Most algorithms will do post pruning
  + Quality-wise: Post pruning > pre pruning
  + Performance-wise: Post pruning < pre prunin

Bagging and Boosting:

* Bagging
  + A single decision tree is really bad. Singular decision trees are not good at prediction. On the other hand, if you had a bunch of decision trees that are *built differently*, that will get you better performance
* Bootstrapping aggregation (?)
  + Has a bunch of features and rows
  + You will take a subsample of the dataset (Rows and Columns) and create a tree from that
  + Do the same with the other portion left out
  + 
  + You’ll get a bunch of trees, and you make your predictions based on those trees,
  + If ⅔ trees output B and one outputs an R, output is B
  + If 1, 2 ,3, it is the average
  + Each dataset will create a different tree, you’ll get much better predictions this way.
* **Boosting assigns a weight to each of the trees when bagging.**
* To do bagging and/or boosting
  + Random forest
  + Some other algorithms and trees (?)
* Note: trees are different than NN
* Bagging and boosting create a bunch of trees
* **You can use Boosting and Bagging with NN, but this isn’t widely used yet due to performance**
* Quiz on tuesday, make sure to go over the notes, starts at 4:30 ends at 4:35
* We will then talk about evaluation. Everything on the notes is fair game for the quiz.

….

End of recording

**The way the professor makes the quizzes:**

* Create a question bank and upload into webcourses
* Double check the questions
* **For any important concept, a question is created**
* **The notes are generated by jotting down the core ideas**
* **Partial credit on multiple answer (?) questions:**
  + **For every answer you select wrong, you lose credit. Its this way so you don’t select every answer and get them all correct**
* “I’m going to try and avoid mathematical questions on the quiz because they’re time”
* [more likely to have questions about the equations, no calculations]
* **“If you understand all the topics discussed in the notes you should be fine” - Dr. Holland”**
* Every time we finish talking about an algorithm, we’ll have a quiz on it and some homework.