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

# Day 9 (Tuesday, Feb 9):

(?) = missing details to be filled in

**Quiz on Chapter 4 Today**

Recording starts

Admin stuff:

* Homework 1 will be posted this week
  + About 2 weeks
  + Beat the benchmark score
  + “I’ll tell you what models to use [decision trees]”
  + **Due somewhere around Feb. 22nd**
  + I’ll give you two datasets (one with features and one with target vectors)
  + You’ll have to build decision tree, RAND forest, try to train these into the data
    - Get the best model between the two options
    - Once your happy with what you’ve got, submit your model and submit to kaggle.
    - You need to beat a benchmark score [more on that later]
    - Half of the points = beat the benchmark
    - Other half = the link to your notebook where your code is
      * Make sure you have models, output is safe, see you generate stuff etc etc [more on that later]
    - If you just make a csv file without writing the code, you will get a 50%
    - This assignment is an individual assignment
* Datacamp
  + You should be in about 4-5 courses at this point
* Remember that Datacamp courses give you extra credit

Next week:

* Evaluation techniques on how to measure performance/quality of your algorithm

Chapter 9

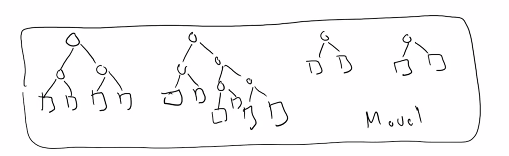
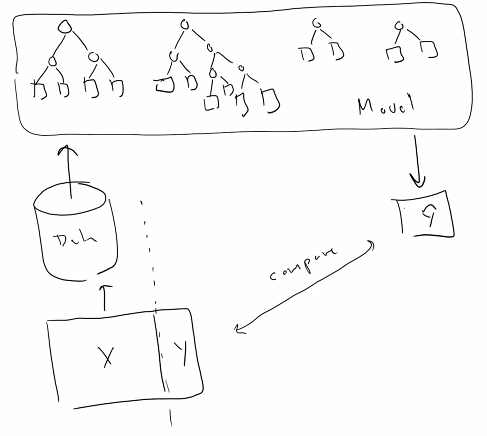
Evaluation:

Why are we skipping to chapter 9?

* Reason skipping from chapter 4 - 9 is because we have a (???)
* You can all build a model given some data that will give some prediction
* We need to figure out how good of a prediction our generated prediction is
* We need a way to figure out how to evaluate our algorithms and figure out if they’re any good

We are going to go in the order of the slides when we cover this stuff

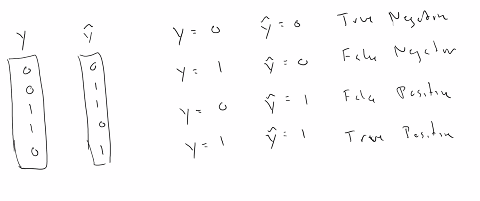
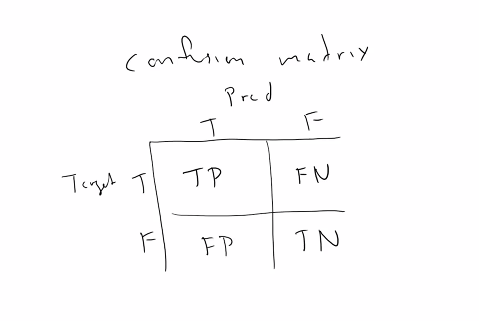
Scenario:

* We may have a decision tree, or a “forest” (when you bag the data and create a bunch of decision trees from the subset)
* 
* This is our model.
* What does our model take as input?
  + Data
* What does our model output?
  + “Predictions! Some sort of answer”
  + “Thats great, thats useful, what do we have, what do we know with this data. When we are training our model initially, what does our dataset consist of?”
  + Its got features and the answers
* The way that we figure out how well this model does is that we compare the known answers with the predictions from the model
* 

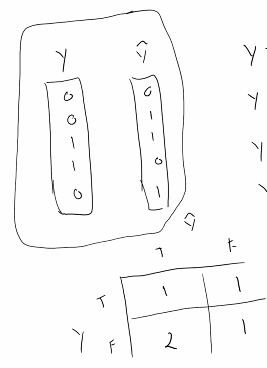
How do we compare our predicted values with a set of known answers?

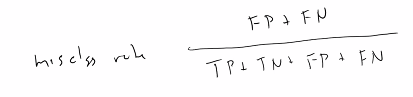
* One of the things that we might want to ask, given the known/given vectors is to calculate a percentage, a misclassification rate

Misclassification rate:

* A function that takes in the known values and the predicted values and returns a percentage describing how many elements were misclassified
* # wrong / total
* Does this make sense in terms of a regression problem?
  + No
* When we are talking about a classification problem, the predictions that we make are discrete variables. FOr simplicity sake, we assume that our target vector is binary
* Y dataset values yt = 0 0 1 1 0
* Y predicted values = 0 1 1 0 1
  + We can have a few scenarios:
  + 
    - yt = 0 = 0 [True negative]
    - yt = 1 = 0 [False negative, you falsely predicted negative]
    - yt = 0 = 1 [False positive, you falsely predicted positive]
    - yt = 1 = 1 [True positive, you falsely predicted negative]
  + False negatives and false positives are serious business
  + **The way that you will see these a lot of times represented, is by using a confusion matrix:**
    - 
  + It is called the confusion matrix because:
    - It is associated with your models being confused
    - Also associated with being confusing to read

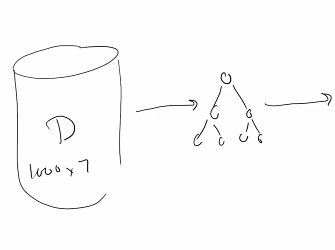
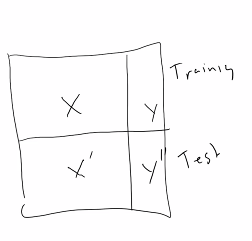
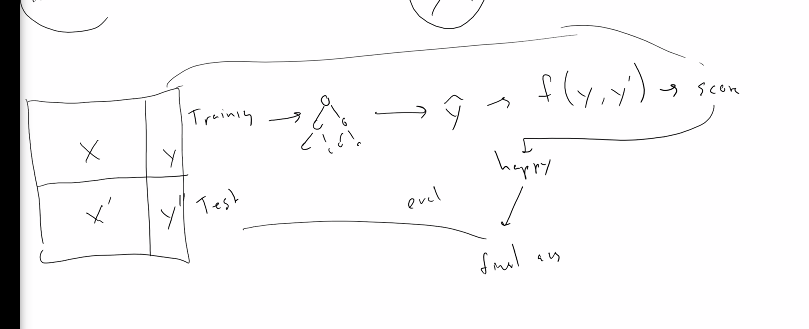
Looking at this dataset, create the confusion matrix:

* Y dataset values y = 0 0 1 1 0
* Y predicted values = 0 1 1 0 1
* This is usually done using a .count method
* 
* 1 true positive, 2 false positive, 1 true negative, 1 false negative



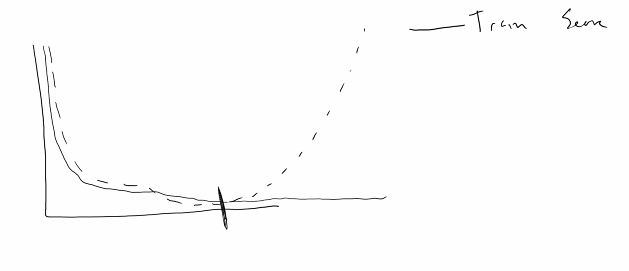
* misclassification rate
* From our predictions using the decision tree and comparing our predictions to the dataset using a confusion matrix, we can compare our decision trees.

What will happen if we make a decision tree based on the entire dataset?

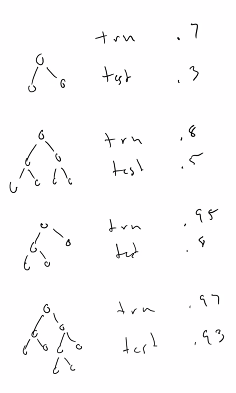
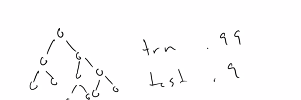
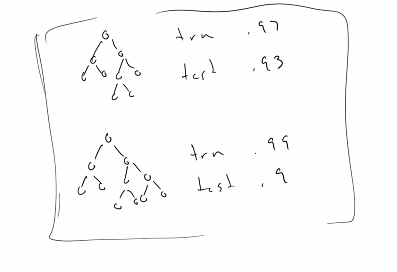
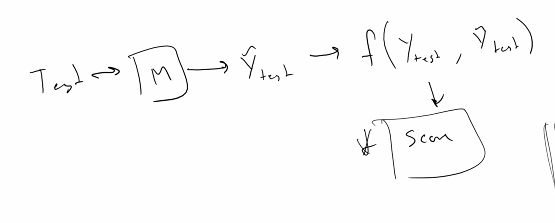
* Let's say we have dataset D() which has 1000 rows in it, and 7 features
* We want to build a tree on this model.
* We evaluate the features, calculate entropy etc etc find the best features to gen nodes from….
* 
* “I bet I can make a decision tree with a misclassification rate of 0”
* It can have one leaf for every single row, and it will have a misclassification rate of 0
  + Is that a problem?
    - Most likely
* How can I prevent this from happening? [overfitting]
* Pruning is good for a decision tree, not a regression model
* We will take this dataset, and we will cut the rows:
* 
* One will be the training set, and one will be the test set
* We use the training set on our model
* <https://learn.datacamp.com/courses/supervised-learning-with-scikit-learn>
* Once we are happy with it, we will evaluate it on the test set
* 
* Splitting into a test and training set will let you know if you get a bad answer

The process:

1. Build model on the training data
2. Evaluate on test data

* This process will allow us to create better models. [mainly avoids overfitting]
* As a result, if you plot your evaluation metric, you’ll get a curve
* Essentially what happens is that your training data:
  + You’ll see that your score on the test data will go down and **after a point, it will go back up as it over fits**
  + ****
* In the case of a decision tree, when we talk about other models, in the upcoming weeks, like linear models:
  + Your linear model has a test associated with it
* The decision tree is a nonparametric model (?)
* This means that your dataset does not have a training metric associated withit
* A parametric model, the number of parameters are fixed. This will make more sense later
* Regarding the decision tree, you need to pick the decision tree at the lowest point on the graph above

Example:

* Lets say you have a training model that evaluates to a training error of .7 (?)
* The test data evaluates to a training error of .3
* Build another tree, it might get better or worse:
* 
* 
* Notice that the test error goes down with the last iteration, therefore our best decision tree is between the last two decision trees
* 
* The problem with having two sets of test data, is that the model will **bias** the model (?)
* The score that we will give for the final test performance is not necessary how well the model is going to generate things. The whole reason that we are doing this is so that we can generalize for data we cannot see. We can then trust that those predictions are correct
* When we are using data to inform the decisions in our tree, we get some sort of (?)
* The way that you get around that problem is “two sets of data are a (?), three sets are great”
* So, we will create *another split* in the dataset:
  + Training
  + Validation
  + Test
* We run the process we described earlier,
* Training the model with the training data,
* Test using the test data, create a model
* [new] Evaluate on the validation data at the very end
* Once you get your prediction of the test data, you then feed that into your performance metric.
* 
* Lets say you don’t like your results:
  + You have to resegment your data entirely and redo the process over from scratch
* We know you need 3 datasets, but how do you determine what goes into those 3 datasets?

Tangent- How do we know if a score is good:

* Generally speaking, if you randomly guess, your missclassifcation rate should be around .5
* The rock curve - looks at false pos/negs and ends up to about .5 for random
* If you guess, you’ll have a baseline
* **Depending on your organization, if you can make a 1% difference, that oculd be a $1 Billion increase**
* In the academic setting, this is usually not the case.
* From the previous research on this dataset, you can know how well everyone has done and you can try to do better than them
* If you are working on a novel problem (every problem in the real world)
  + Find out how well the correct system works and go for anything better
  + You amy find that the current system for…
    - “Find out whether a plane takes off on time”
    - If the current method is just some dude who tracks the planes and takes the average, and makes judgements based on that…
    - If you can build a ML algorithm that runs better than that system, (i.e dude is wrong 60% of the time)
    - If you can do it better than 60%, than its a good model
    - If you can’t, figure out what went wrong
  + The best machine learning algorithms are usually within the 90-95% accuracy rate
  + There are a lot of network architectures that makes all these processes a lot easier.
  + But yeah, in terms of determining if my score is any good depends on the problem
* End of tangent

Bootstrapping:

* One of the ways that you can go between test, validation, and training data is bootstrap sampling
* **Bootstrap sampling** - From your dataset, lets say you have 1000 elements…
  + The training set has 100 elements
  + The validation set has 50 elements
  + You build *once,* and then you get a score
  + Resample
  + Pull out another 100 samples
  + 50 samples
  + Build a new model from scratch
  + You get a score.
  + Repeat x number of times.
  + What you end up with as a result of this, is a bunch of scores and then you average and you get the model score.
  + That gives you some indication of how your model performs
  + One of the ways you can pick the model to use is to keep the one with the highest score
  + Another way is to (?)
* Recap of bootstrap modeling:
  + Every model has hyper parameters,
    - In case of decision tree thats like maximum depth, number of leaf nodes, etc etc
  + Basically train your model 1000 times, don’t throw away the model like bootstrap sampling (?)
  + After training completes, adjust parameters
  + Keep doing it
  + If results unsatisfactory, will have to do whole thing over again
  + More on this thursday

Out of time

Things to be discussed:

* Quiz answers
* Will talk about cross validation
* Bootstrap sampling
* Other performance networks