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

(?) <- if u see this, it means there is some information lost due to lag or lost focus

# Day 11 (Tuesday, Feb 16):

Prelecture:

* **Quiz this Thursday**
* Talk about random stuff
  + Tea
  + No power in texas
  + Benchmark for hw1 is staying put
  + Your training model should be high
  + There isn’t enough time to talk in detail about exact implementation of ML techniques in this course, that’s up to you to figure out, as you are computer science majors after all
  + Datacamp

Recording starts:

Assignment 1:

Recommended courses for getting into assignment 1:

<https://learn.datacamp.com/career-tracks/machine-learning-scientist-with-python>

<https://learn.datacamp.com/courses/machine-learning-for-finance-in-python>

<https://learn.datacamp.com/courses/machine-learning-with-tree-based-models-in-python>

<https://learn.datacamp.com/courses/dealing-with-missing-data-in-python>

<https://learn.datacamp.com/courses/supervised-learning-with-scikit-learn>

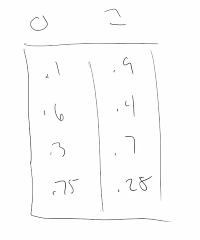
* Assignment 1 has been posted
* **If you’re lost on Kaggle**, check out <https://www.kaggle.com/c/titanic>
* **If you’re lost on SciKit,** check out <https://sklearn.org/documentation.html>
* Recommends you look at Datacamp tutorials regarding Machine learning if you are lost
  + More specifically, the **machine learning track** to learn the general idea for ML implementations
    - <https://learn.datacamp.com/career-tracks/machine-learning-for-everyone>
  + The **data science course for everyone** to learn about cleaning your data
    - <https://learn.datacamp.com/career-tracks/data-scientist-with-python>
  + There are a few other courses, but there is a “lot of overlap” between the courses
* If you’re worrying about your grade, remember you get 1% to your final grade for 10 courses you complete on datacamp

Regarding Assignment 1’s content itself:

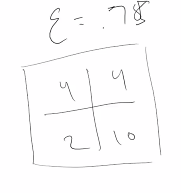
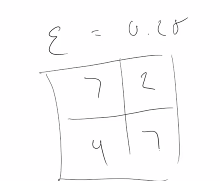
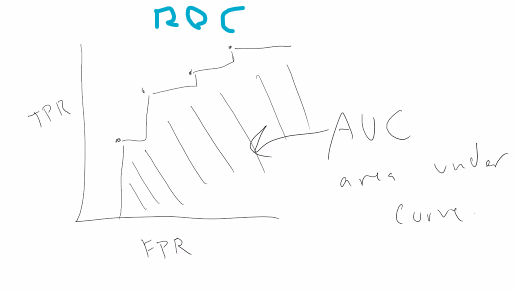
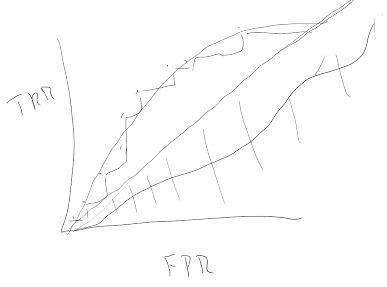
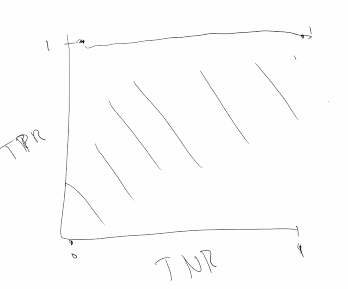
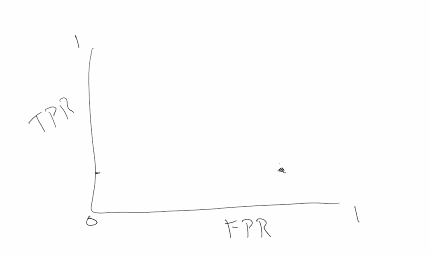
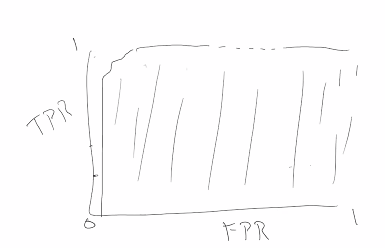
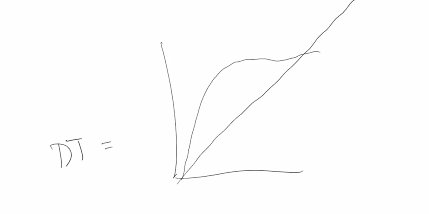
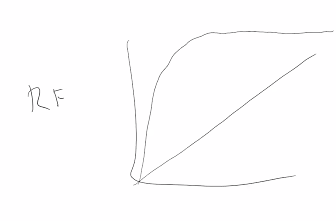
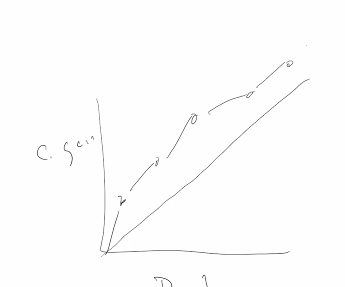
* **Decision trees and Random Forests only**
* You’re deciding whether or not a company is going to go bankrupt
* Best to keep the notebook private UNTIL the due date
* You can host the notebook on either Kaggle or Google Colab
* If your score is under the benchmark, you gotta change it, and also make sure you’re using the right function to make your predictions [probability based]
* Half your grade:
  + Is based on whether or not your model passes the benchmark
  + Two people having exactly the same score is suspicious
  + Output is a probability between 0 and 1.0
* The other half of your grade:
  + Is based on the content of your notebook
  + Make sure you do everything listed in the notebook requirements
  + **The notebook must contain both a DecisionTree and a RandomForest**
  + **Make sure your notebook displays output for every cell** (if its just a bunch of import statements in the cell and it doesn’t print out any output that is fine)
* “Everybody needs to struggle with this stuff for a little bit”
* Note:
  + There are two datasets, test.csv and train.csv
  + Build your model on train.csv, as it contains all the features as well as the target feature, “Bankrupt”
  + Then, yourModel.somePredictionFunction(train.csv) and return some probability of going bankrupt
  + To find that documentation:
    - [https://scikit-learn.org/stable/modules/classes.html](https://scikit-learn.org/stable/modules/classes.html#regression-metrics)
    - Highly recommend looking at Datacamp before looking at link above, as once you do some tutorials, you’ll know where to look

Moving on…

More stuff on Evaluation:

* [Recap] Confusion matrix
  + True/false postive/negative, 4 cell grid
  + Used to find the other metrics
* [Recap] Precision/Recall
  + Precision - (?)
  + Recall - how many correct predictions we have made
  + Accuracy - how many you’ve gotten right on all the dataset
* [Recap] Profit/loss matrices
  + Basically just Confusion matrices, just emphasizes (through weights) the true positive/negative
* Most models don’t give you the answer, they just give you teh probability of something being the answer
* Lets say we have a model on some dataset:
  + It spits out .9, .4, .3, .75
  + 
  + These numbers are the probability of one thing in the dataset (?)
  + Most machine models give you sort of a probability that y takes on a certain value, that is the prediction score
  + It so happens that the performance metric to calculate the area under the curve requires you to feed it a score, not a class
  + There is one prediction within scikit learn gives you the class
  + There is another one that will give you the probability of a certain class
  + The function for assignment 1 can be found within the API documentation
  + Note that you can treat a binary problem as a 2 class problem as the models return a probability:
  + 

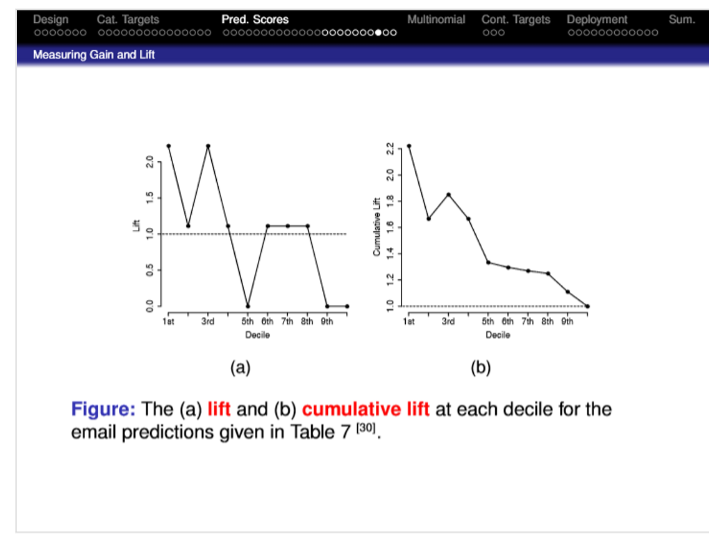
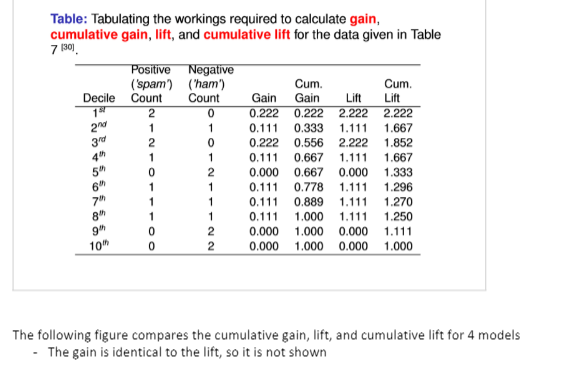
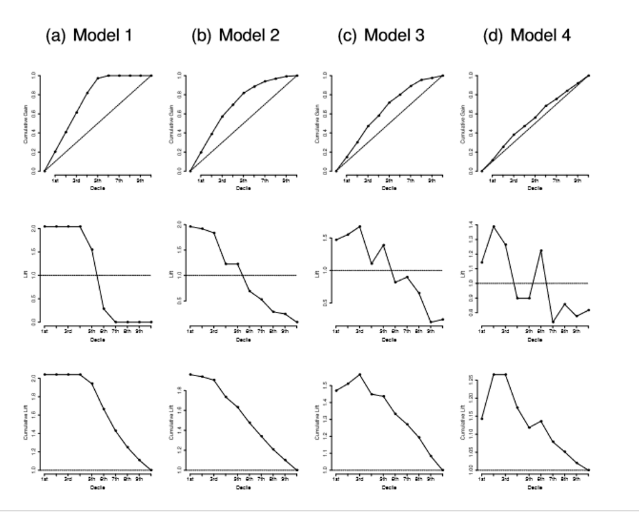
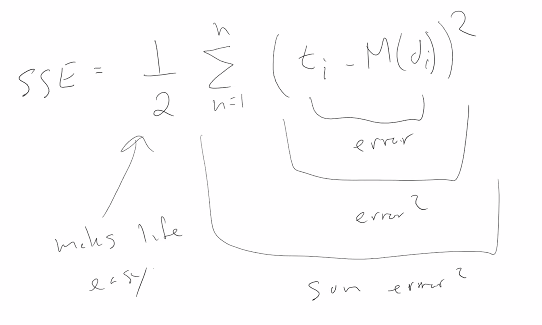
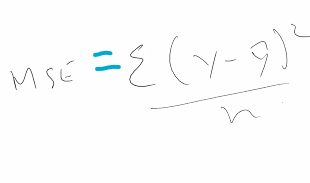
The receiver operating characteristic:

* Called the ROC curve, or just R O C,
* Additionally, there is the ROC\_AUC (Receiver Operating Characteristic Area Under the Curve)
* Example:
  + Spam vs Ham
  + Anything that the machine returns that returns a score of .75 or less:
  + We might see this confusion matrix
  + 
  + If we change the epsilon value to .28, we might see this:
  + 
  + By changing the threshold, we can change the true positive/negative rates
  + The notion is that:
    - We generate a confusion matrix for epsilon = [0, 1]
    - With some level of discretization (0.1, 0.2, 0.3, 0.4, 0.5….)
    - We will then get different true positive/negative rates
    - We can graph this, with y = TPR x = FPR
  + Graph of ROC:
  + 
  + If we generate more points on this graph, eventually we’ll get something that looks like a curve
  + Generally speaking, you’ll see a line that goes in a 45 deg angle, **This is the line you get for random selection**
  + ****
  + So what does this represent?
    - (?)
  + What you want in an ROC curve:
    - A perfect model, no false positives
    - 
    - Best\_Score(ROC\_AUC) = 1
    - Worst\_Score(ROC\_AUC) = 0
  + Basically, at what value, do I consider my prediction to be positive?
    - Goes into the notion that you feed in data, and it spits out some probability,
    - Every epsilon, every threshold you want to pick is on the ROC plot.
    - It represents the TPR and TNR generated by your model.
    - So, you choose your model
    - Choose your TPR and TNR rates
    - 
    - We can graph the probability outputted from the model as a point on the AUC curve after defining our rates (our epsilons):
    - 
    - We can mess around with our epsilon values,
    - [tangent] The way that the actual ROC scoring mechanism works:
      * It says “i’m going to generate a list of a thousand epsilon values, I’m going to calculate the TPR and TNR for each of the epsilons and curve it”
      * The epsilon, the threshold value is the probability is if it is true.
      * If your threshold is really high, everything is going to be considered as a positive
      * “That’s kinda why it increases, everything is positive”
      * ROC represents tradeoff of the TPR and FPR rates, in a real implementation “your model is always right”
      * What you hope to gain is basically a model that goes up and then has a little bit of activity liek this:
      * 
      * This is the 97-98 scores for AOC
      * **So, if your doing an ROC curve, you aren’t trying to find an epsilon value**, you just want to evaluate if your model is any good
    - So your DecisionTree model may give you a curve that looks like this,
    - 
    - While your RandomForest may get you something like this:
    - 
    - In this case the DecisionTree.area < RandomForest.area
    - The points on these plots represents a tradeoff, you sacrifice TPR or TNR for one or the other
  + “The main thing you need to know
    - Higher ROC scores are better, and there are libraries that do that for you.
    - You just need to feed in your model or a list of prediction scores, and will cut at different points to generate the different curves
    - If you want to write your own, thats a different story”
  + So there is another metric you may see:
    - KS statistic- a useful metric for comparing distributions, “I’m going to let you read about that one”
  + There is a concept that appears on interview questions:
    - “Gain and Lift”
    - When you are doing your performanc metrics, its a question of choices
    - Do you care as positive cases being True positives?
    - Do you care more about negative cases being True negatives?
    - Area under ROC
    - Gain and lift are used when you are concerned about predicting the positive instances, not how well it separates the classes.
      * Sometimes we are ok with it getting things wrong, except for the case of positive instances
    - Gain and Life (Do you mean Lift) -
      * Take ur dataset
      * Cut it up into 10 parts called Depth size (tensile?)
      * This cut is done by sorting on the prediction score then cutting
      * You’re looking for
        + “How do these components work if things were random?”
      * The **gain** = Number of positives in each tensile / total number of positives = [0, 1]
      * **Cumulative gain - “**Number of positives in and under the current decile” / total number of positives
      * **“**Number of positives in and under the current decile” = if you’re looking at decile 5, then its the number of positives in 1,2,3,4,5.
      * Graph looks something like this:
      * 
      * Lift: how much higher the positive cases for an (?? too fast) expected rate. From the notes it says “The lift tells us how much higher the actual percentage of positive instances in a decilel is than the rate expected”
        + Higher lift value indicate the model is doing well; a lift of 1 indicates the model is performing as if it was random.
        + Lift = percentages of positive cases in decile / percentage of positives = [0, infinity)

**Lift(dec)**=

* + - * + Cumulative Lift = percent positive up to decile / percent positive
        + **Cumulative Lift(dec)**=

The cumulative lift can also be calculated but notice that it decreases as you move right, approaching 1.0

* + - * + 
        + 
        + 
      * All these metrics can be applied to multiple classes (we have not talked about that, but we will get there)
      * We will take a confusion matrix if we have 5 instead of 2
      * We can calculate a confusion matrix for that,
* If you have a continuous target:
  + You are trying to predict a continuous number
  + Remember the R^2 value from statistics
  + R^2 = 1 - sum of square errors / total sum of squares
  + Where the sum of square errors == ½ Summation(n=1, n, ti - M(di))^2
    - sum of squared errors =
  + 
  + The ½ is there due to differentiations, it makes life easy, it cancels some stuff out, its a scaling factor
  + R^2 is used to measure variation of a model,
  + Mean Square Error (MSE) is just the summation of the error
    - Mean Square Error =
    - 
  + That's all we got to talk about for evaluation
  + The main thing to focus on:
    - Notes have stuff at the end about modeling distribution - useful and important to know, **”I don’t know how many quiz questions are on that” [this implies something covered in notes and not in lecture?]**

Main Takeaway:

* + Calculate your Confusion matrix, and you can get a bunch of different metrics for evaluating your model, each with their own use case
  + There are generally more options for classification

Recording stops

“Office hours” begin

Note: if you standardize the data, make sure that you manipulate both the training data and the test data the exact same way