Columbia Data Science course, week 12: Predictive modeling, data leakage, model evaluation

November 20, 2012 Cathy O'Neil, mathbabe

This week's guest lecturer in Rachel Schutt's Columbia Data Science class was Claudia Perlich. Claudia has been the Chief Scientist at m6d for 3 years. Before that she was a data analytics group at the IBM center that developed Watson, the computer that won Jeopardy!, although she didn't work on that project. Claudia got her Ph.D. in information systems at NYU and now teaches a class to business students in data science, although mostly she addresses how to assess data science work and how to manage data scientists. Claudia also holds a masters in Computer Science.

Claudia is a famously successful data mining competition winner. She won the KDD Cup in 2003, 2007, 2008, and 2009, the ILP Challenge in 2005, the INFORMS Challenge in 2008, and theKaggle HIV competition in 2010.

She's also been a data mining competition organizer, first for the INFORMS Challenge in 2009 and then for the Heritage Health Prize in 2011. Claudia claims to be retired from competition.

Claudia's advice to young people: pick your advisor first, then choose the topic. It's important to have great chemistry with your advisor, and don't underestimate the importance.

Background

Here's what Claudia historically does with her time:

- predictive modeling
- data mining competitions
- publications in conferences like KDD and journals
- talks
- patents
- teaching
- digging around data (her favorite part)

Claudia likes to understand something about the world by looking directly at the data.

Here's Claudia's skill set:

- plenty of experience doing data stuff (15 years)
- data intuition (for which one needs to get to the bottom of the *data generating process*)
- dedication to the evaluation (one needs to cultivate a good sense of smell)
- model intuition (we use models to diagnose data)

Claudia also addressed being a woman. She says it works well in the data science field, where her intuition is useful and is used. She claims her nose is so well developed by now that she can smell it when something is wrong. This is not the same thing as being able to prove something algorithmically. Also, people typically remember her because she's a woman, even when she don't remember them. It has worked in her favor, she says, and she's happy to admit this. But then again, she is where she is because she's good.

Someone in the class asked if papers submitted for journals and/or conferences are blind to gender. Claudia responded that it was, for some time, typically double-blind but now it's more likely to be one-sided. And anyway there was a cool analysis that showed you can guess who wrote a paper with 80% accuracy just by knowing the citations. So making things blind doesn't really help. More recently the names are included, and hopefully this doesn't make things too biased. Claudia admits to being slightly biased towards institutions – certain institutions prepare better work.

Skills and daily life of a Chief Data Scientist

Claudia's primary skills are as follows:

- Data manipulation: unix (sed, awk, etc), Perl, SQL
- Modeling: various methods (logistic regression, nearest neighbors, k-nearest neighbors, etc)
- Setting things up

She mentions that the methods don't matter as much as *how you've set it up*, and how you've translated it into something where you can solve a question.

More recently, she's been told that at work she spends:

- 40% of time as "contributor": doing stuff directly with data
- 40% of time as "ambassador": writing stuff, giving talks, mostly external communication to represent m6d, and
- 20% of time in "leadership" of her data group

At IBM it was much more focused in the first category. Even so, she has a flexible schedule at m6d and is treated well.

The goals of the audience

She asked the class, why are you here? Do you want to:

- become a data scientist? (good career choice!)
- work with data scientist?
- work for a data scientist?
- manage a data scientist?

Most people were trying their hands at the first, but we had a few in each category.

She mentioned that it matters because the way she'd talk to people wanting to become a data scientist would be different from the way she'd talk to someone who wants to manage them. Her NYU class is more like how to manage one.

So, for example, you need to be able to evaluate their work. It's one thing to check a bubble sort algorithm or check whether a SQL server is working, but checking a model which purports to give the probability of people converting is different kettle of fish.

For example, try to answer this: how much better can that model get if you spend another week on it? Let's face it, quality control is hard for *yourself* as a data miner, so it's *definitely* hard for other people. There's no easy answer.

There's an old joke that comes to mind: What's the difference between the scientist and a consultant? The scientists asks, how long does it take to get this right? whereas the consultant asks, how right can I get this in a week?

Insights into data

A student asks, how do you turn a data analysis into insights?

Claudia: this is a constant point of contention. My attitude is: I like to understand something, but what I like to understand isn't what *you'd* consider an insight. My message may be, hey you've replaced every "a" by a "0", or, you need to change the way you collect your data. In terms of useful insight, Ori's lecture from last week, when he talked about causality, is as close as you get.

For example, decision trees you interpret, and people like them because they're easy to interpret, but I'd ask, why does it look like it does? A slightly different data set would give you a different tree and you'd get a different conclusion. This is the *illusion* of understanding. I tend to be careful with delivering strong insights in that sense.

For more in this vein, Claudia suggests we look at Monica Rogati's talk "Lies, damn lies, and the data scientist."

Data mining competitions

Claudia drew a distinction between different types of data mining competitions.

On the one hand you have the "sterile" kind, where you're given a clean, prepared data matrix, a standard error measure, and where the features are often anonymized. This is a pure machine learning problem.

Examples of this first kind are: KDD Cup 2009 and 2011 (Netflix). In such competitions, your approach would emphasize algorithms and computation. The winner would probably have heavy machines and huge modeling ensembles.

On the other hand, you have the "real world" kind of data mining competition, where you're handed raw data, which is often in lots of different tables and not easily joined, where you set up the model yourself and come up with task-specific evaluations. This kind of competition simulates real life more.

Examples of this second kind are: KDD cup 2007, 2008, and 2010. If you're competing in this kind of competition your approach would involve understanding the domain, analyzing the data, and building the model. The winner might be the person who best understands how to tailor the model to the actual question.

Claudia prefers the second kind, because it's closer to what you do in real life. In particular, the same things go right or go wrong.

How to be a good modeler

Claudia claims that data and domain understanding is the single most important skill you need as a data scientist. At the same time, this can't really be taught – it can only be *cultivated*.

A few lessons learned about data mining competitions that Claudia thinks are overlooked in academics:

- Leakage: the contestants best friend and the organizers/practitioners worst nightmare. There's always something wrong with the data, and Claudia has made an artform of figuring out how the people preparing the competition got lazy or sloppy with the data.
- Adapting learning to real-life performance measures beyond standard measures like MSE, error rate, or AUC (profit?)
- Feature construction/transformation: real data is rarely flat (i.e. given to you in a beautiful matrix) and good, practical solutions for this problem remains a challenge.

Leakage

Leakage refers to something that helps you predict something that isn't fair. It's a huge problem in modeling, and not just for competitions. Oftentimes it's an artifact of reversing cause and effect.

Example 1: There was a competition where you needed to predict S&P in terms of whether it would go up or go down. The winning entry had a AUC (area under the ROC curve) of 0.999 out of 1. Since stock markets are pretty close to random, either someone's very rich or there's something wrong. There's something wrong.

In the good old days you could win competitions this way, by finding the leakage.

Example 2: Amazon case study: big spenders. The target of this competition was to predict customers who spend a lot of money among customers using past purchases. The data consisted of transaction data in different categories. But a winning model identified that "Free Shipping = True" was an excellent predictor

What happened here? The point is that free shipping is an *effect* of big spending. But it's not a good way to model big spending, because in particular it doesn't work for new customers or for the future. Note: timestamps are weak here. The data that included "Free Shipping = True" was simultaneous with the sale, which is a no-no. We need to only use data from beforehand to predict the future.

Example 3: Again an online retailer, this time the target is predicting customers who buy jewelry. The data consists of transactions for different categories. A very successful model simply noted that if sum(revenue) = 0, then it predicts jewelry customers very well?

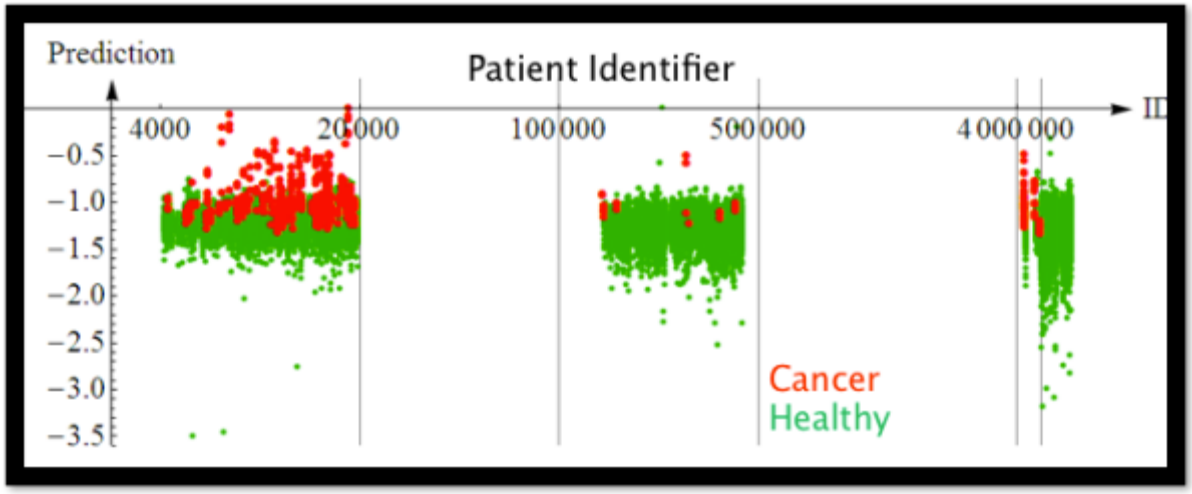
What happened here? The people preparing this data removed jewelry purchases, but only included people who bought something in the first place. So people who had sum(revenue) = 0 were people who only bought jewelry. The fact that you only got into the dataset if you bought something is weird: in particular, you wouldn't be able to use this on customers before they finished their purchase. So the model wasn't being trained on the right data to make the model useful. This is a sampling problem, and it's common.

Example 4: This happened at IBM. The target was to predict companies who would be willing to buy "websphere" solutions. The data was transaction data + crawled potential company websites. The winning model showed that if the term "websphere" appeared on the company's website, then they were great candidates for the product.

What happened? You can't crawl the *historical* web, just today's web.

Thought experiment

You're trying to study who has breast cancer. The patient ID, which seemed innocent, actually has predictive power. What happened?



In the above image, red means cancerous, green means not. it's plotted by patient ID. We see three or four distinct buckets of patient identifiers. It's very predictive depending on the bucket. This is probably a consequence of using multiple databases, some of which correspond to sicker patients are more likely to be sick.

A student suggests: for the purposes of the contest they should have renumbered the patients and randomized.

Claudia: would that solve the problem? There could be other things in common as well.

A student remarks: The important issue could be to see the extent to which we can figure out which dataset a given patient came from based on things besides their ID.

Claudia: Think about this: what do we want these models for in the first place? *How well can you predict cancer?*

Given a new patient, what would you do? If the new patient is in a fifth bin in terms of patient ID, then obviously don't use the identifier model. But if it's still in this scheme, then maybe that really is the best approach.

This discussion brings us back to the fundamental problem that we need to know what the purpose of the model is and how is it going to be used in order to decide how to do it and whether it's working.

Pneumonia

During an INFORMS competition on pneumonia predictions in hospital records, where the goal was to predict whether a patient has pneumonia, a logistic regression which included the number of diagnosis codes as a numeric feature (AUC of 0.80) didn't do as well as the one which included it as a categorical feature (0.90). What's going on?

This had to do with how the person prepared the data for the competition:

icd1x	icd2x	icd3x	icd4x
786	285	459	-1
401	486	-1	-1
401	486	780	-1
599	-1	-1	-1
V22	650	-1	-1
V56	492	586	-1
786	493	285	459

icd1x	icd2x	icd3x	icd4x
786	285	459	-1
401	-1	-1	-1
401	780	-1	-1
599	-1	-1	-1
V22	650	-1	-1
V56	492	586	-1
786	493	285	459

The diagnosis code for pneumonia was 486. So the preparer removed that (and replaced it by a "-1") if it showed up in the record (rows are different patients, columns are different diagnoses, there are max 4 diagnoses, "-1" means there's nothing for that entry).

Moreover, to avoid telling holes in the data, the preparer moved the other diagnoses to the left if necessary, so that only "-1"'s were on the right.

There are two problems with this:

1. If the column has only "-1"'s, then you know it started out with only pneumonia, and
2. If the column has no "-1"'s, you know there's no pneumonia (unless there are actually 5 diagnoses, but that's less common).

This was enough information to win the competition.

Note: winning competition on leakage is easier than building good models. But even if you don't explicitly understand and game the leakage, your model will do it for you. Either way, leakage is a huge problem.

How to avoid leakage

Claudia's advice to avoid this kind of problem:

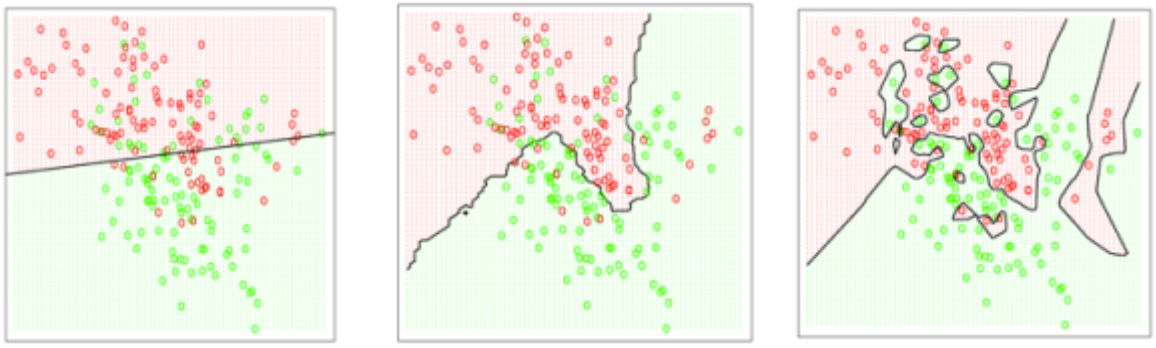
- You need a strict temporal cutoff: remove all information just prior to the event of interest (patient admission).
- There has to be a timestamp on every entry and you need to keep
- Removing columns asks for trouble
- Removing rows can introduce inconsistencies with other tables, also causing trouble
- The best practice is to start from scratch with clean, raw data after careful consideration
- You need to know how the data was created! I only work with data I pulled and prepared myself (or maybe Ori).

Evaluations

How do I know that my model is any good?

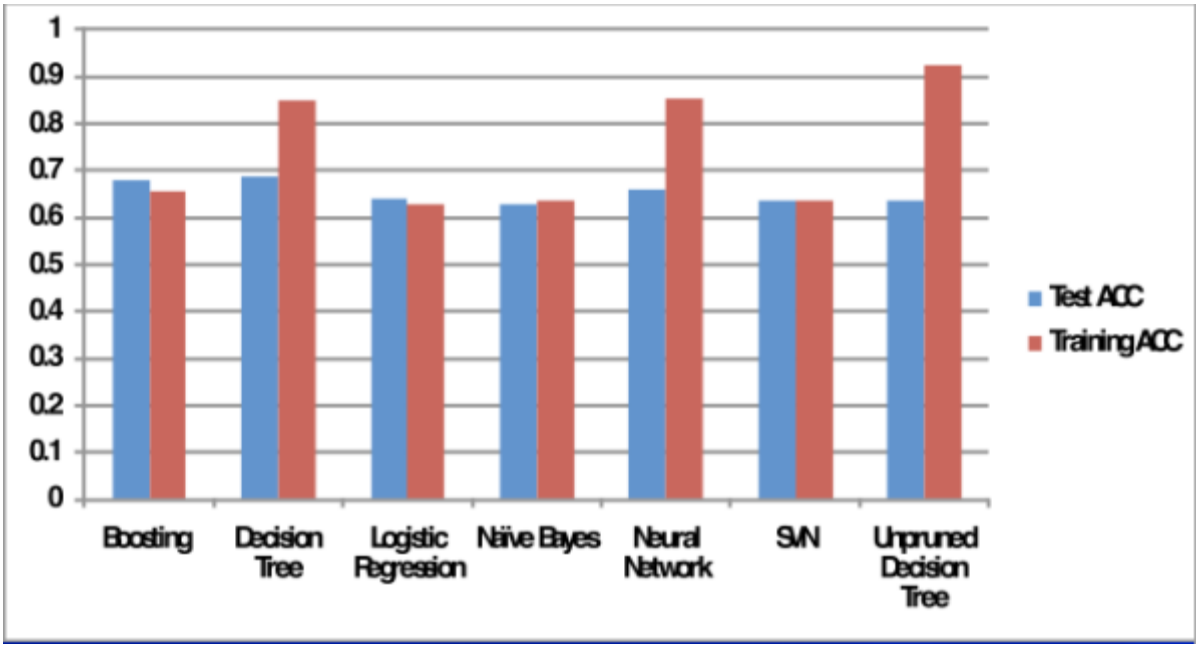
With powerful algorithms searching for patterns of models, there is a serious danger of over fitting. It's a difficult concept, but the general idea is that "if you look hard enough you'll find something" even if it does not generalize beyond the particular training data.

To avoid overfitting, we cross-validate and we cut down on the complexity of the model to begin with. Here's a standard picture (although keep in mind we generally work in high dimensional space and don't have a pretty picture to look at):



The picture on the left is underfit, in the middle is good, and on the right is overfit.

The model you use matters when it concerns overfitting:



So for the above example, unpruned decision trees are the most over fitting ones. This is a well-known problem with unpruned decision trees, which is why people use pruned decision trees.

Accuracy: meh

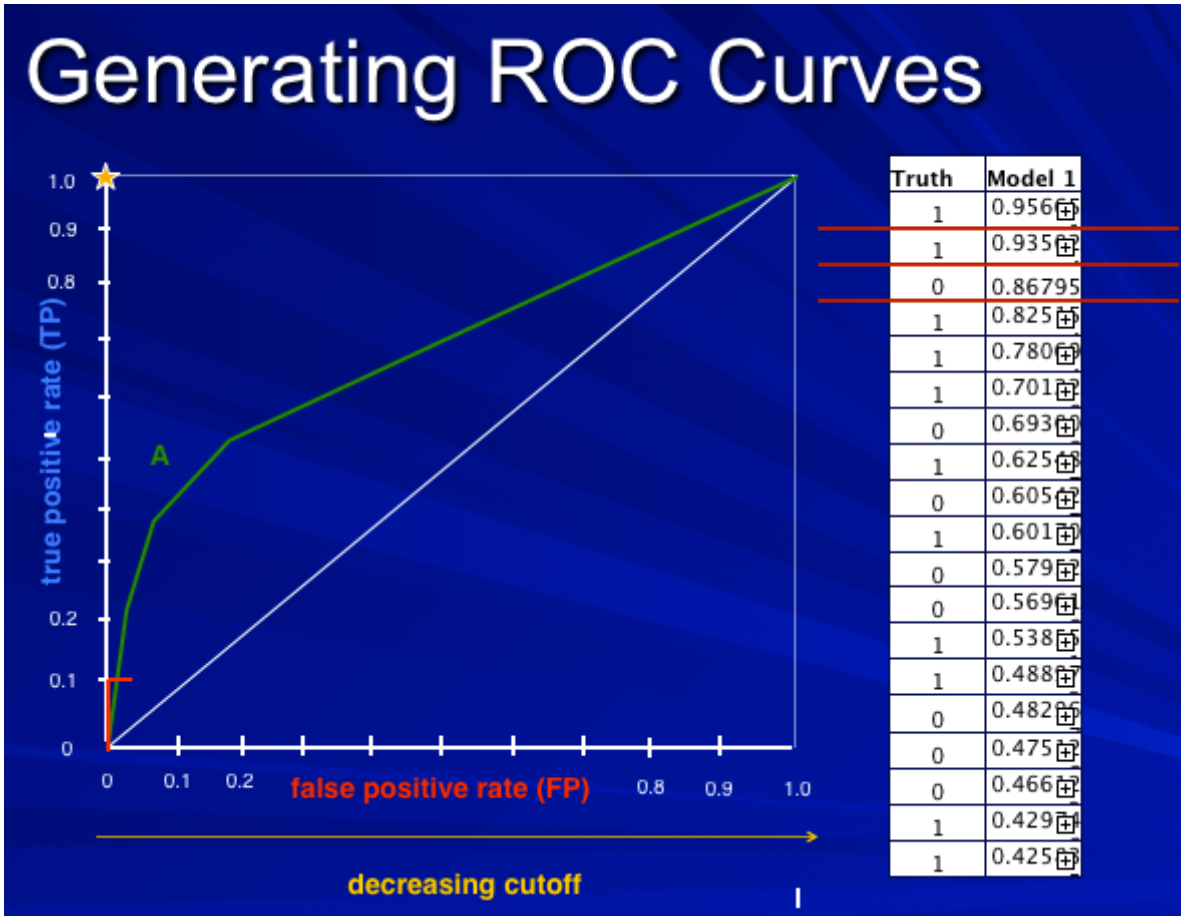
Claudia dismisses accuracy as a bad evaluation method. What's wrong with accuracy? It's inappropriate for regression obviously, but even for classification, if the vast majority is of binary outcomes are 1, then a stupid model can be accurate but not good (guess it's always "1"), and a better model might have lower accuracy.

Probabilities matter, not 0's and 1's.

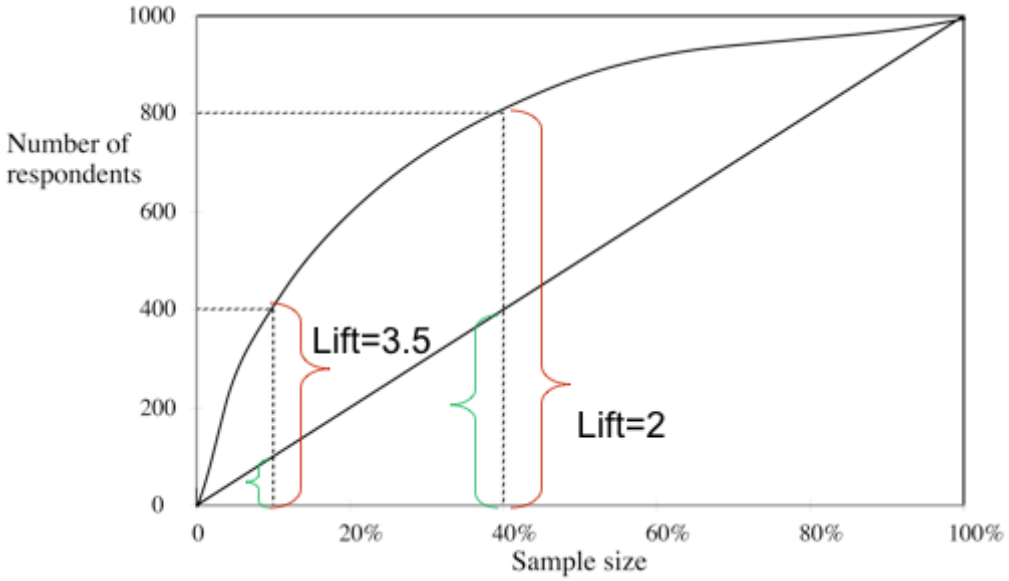
Nobody makes decisions on binary outcomes. I want to know the *probability* I have breast cancer, I don't want to be told yes or no. It's much more information. I care about probabilities.

How to evaluate a probability model

We separately evaluate the ranking and the calibration. To evaluate the ranking, we use the ROC curve and calculate the area under it, typically ranges from 0.5-1.0. This is independent of scaling and calibration. Here's an example of how to draw an ROC curve:



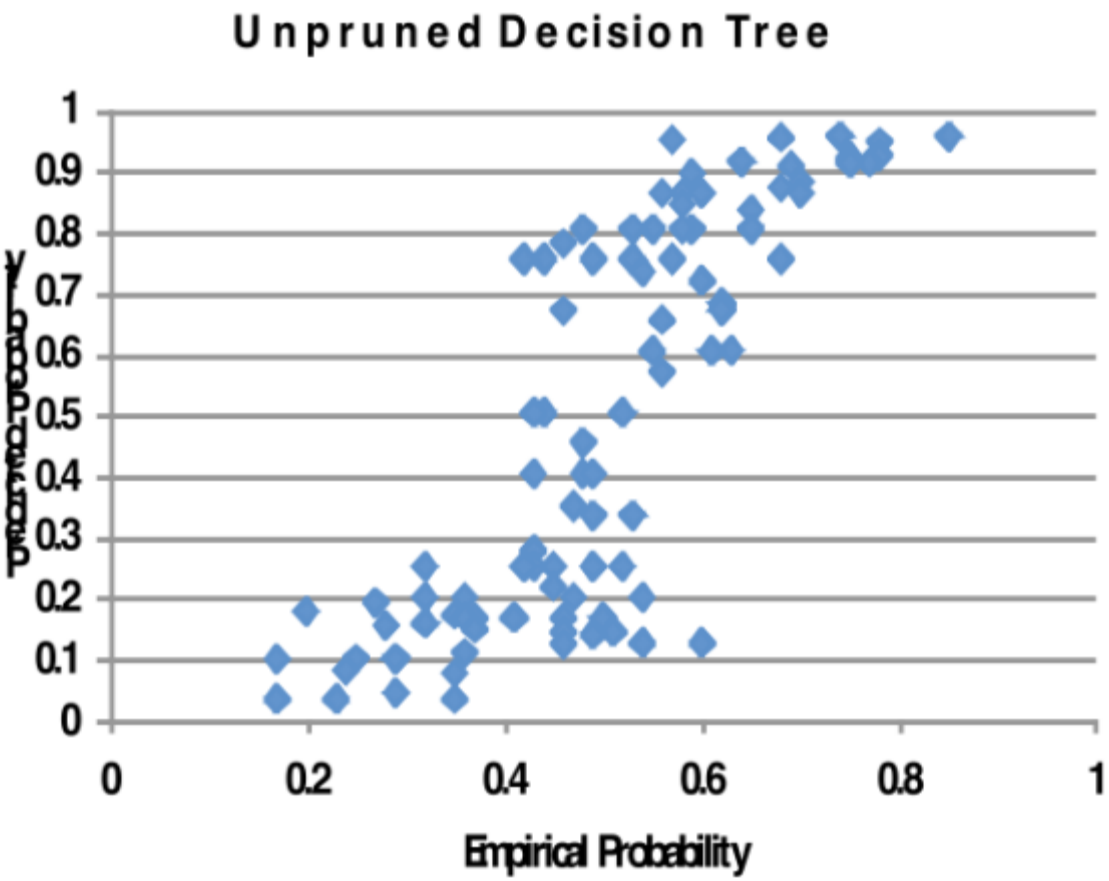
Sometimes to measure rankings, people draw the so-called lift curve:



The key here is that the lift is calculated *with respect to a baseline*. You draw it at a given point, say 10%, by imagining that 10% of people are shown ads, and seeing how many people click versus if you randomly showed 10% of people ads. A lift of 3 means it's 3 times better.

How do you measure calibration? Are the probabilities accurate? If the model says probability of 0.57 that I have cancer, how do I know if it's really 0.57? We can't measure this directly. We can only bucket those predictions and then aggregately compare those in that prediction bucket (say 0.50-0.55) to the actual results for that bucket.

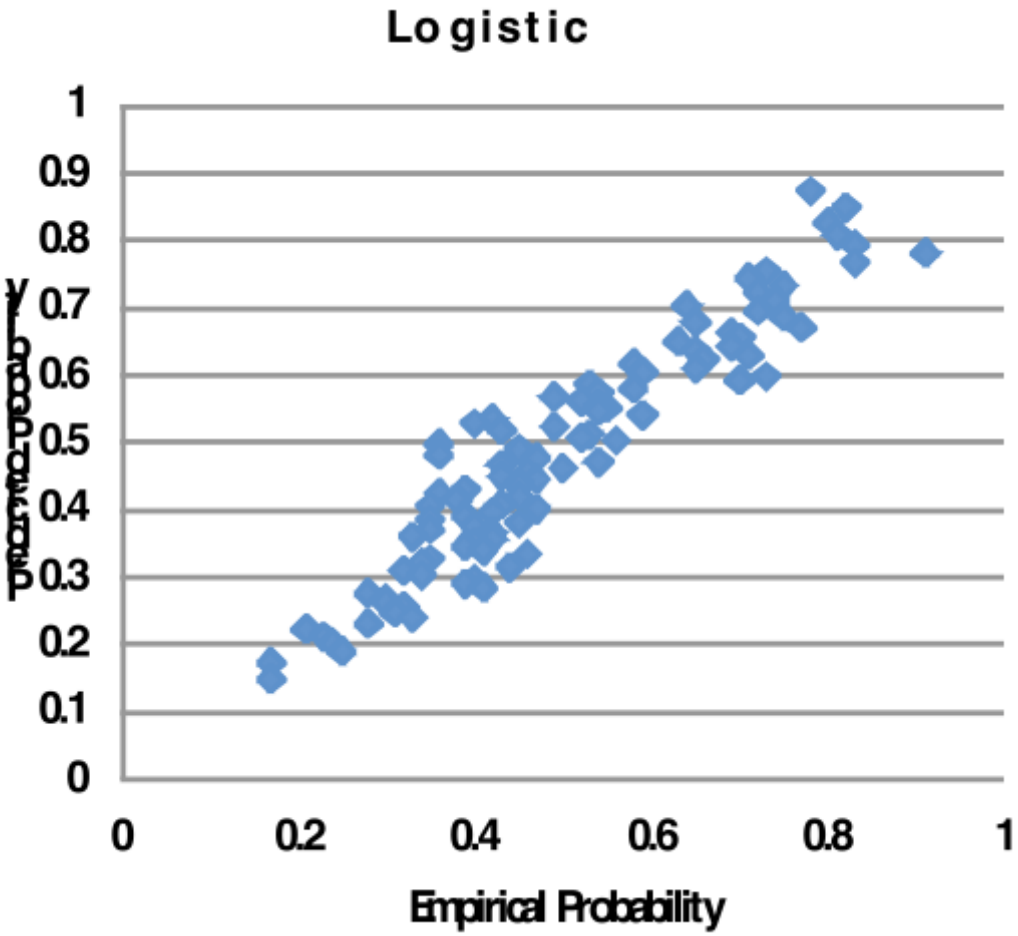
For example, here's what you get when your model is an unpruned decision tree, where the blue diamonds are buckets:



A good model would show buckets right along the x=y curve, but here we're seeing that the predictions were much more extreme than the actual probabilities. Why does this pattern happen for decision trees?

Claudia says that this is because *trees optimize purity*: it seeks out pockets that have only positives or negatives. Therefore its predictions are more extreme than reality. This is generally true about decision trees: they do not generally perform well with respect to calibration.

Logistic regression looks better when you test calibration, which is typical:



Takeaways:

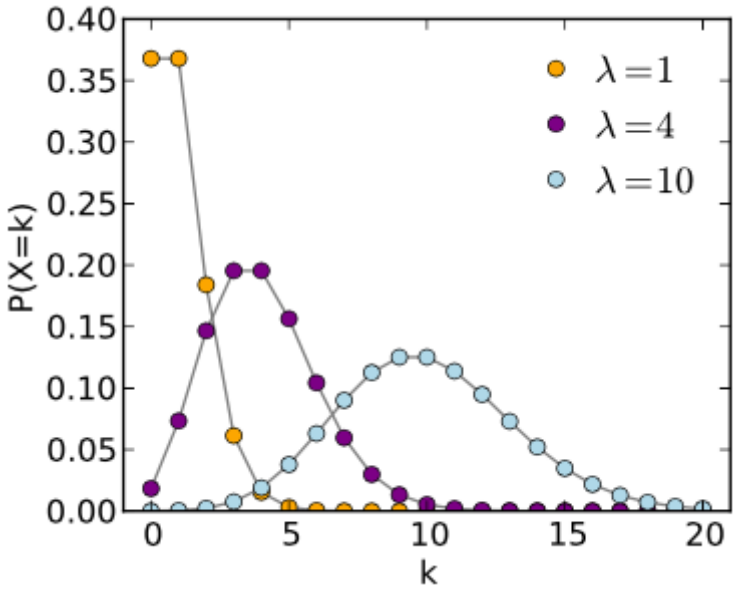
- Accuracy is almost never the right evaluation metric.
- Probabilities, not binary outcomes.
- Separate ranking from calibration.
- Ranking you can measure with nice pictures: ROC, lift
- Calibration is measured indirectly through binning.
- Different models are better than others when it comes to calibration.
- Calibration is sensitive to outliers.
- Measure what you want to be good at.
- Have a good baseline.

Choosing an algorithm

This is not a trivial question and in particular small tests may steer you wrong, because as you increase the sample size the best algorithm might vary: often decision trees perform very well but only if there's enough data.

In general you need to choose your algorithm depending on the size and nature of your dataset and you need to choose your evaluation method based partly on your data and partly on what you wish to be good at. Sum of squared error is maximum likelihood loss function if your data can be assumed to be normal, but if you want to estimate the median, then use absolute errors. If you want to estimate a quantile, then minimize the weighted absolute error.

We worked on predicting the number of ratings of a movie will get in the next year, and we assumed a poisson distributions. In this case our evaluation method doesn't involve minimizing the sum of squared errors, but rather something else which we found in the literature specific to the Poisson distribution, which depends on the single parameter λ:



Charity direct mail campaign

Let's put some of this together.

Say we want to raise money for a charity. If we send a letter to every person in the mailing list we raise about $9000. We'd like to save money and only send money to people who are likely to give – only about 5% of people generally give. How can we do that?

If we use a (somewhat pruned, as is standard) decision tree, we get $0 profit: it never finds a leaf with majority positives.

If we use a neural network we still make only $7500, even if we only send a letter in the case where we expect the return to be higher than the cost.

This looks unworkable. But if you model is better, it's not. A person makes two decisions here. First, they decide whether or not to give, then they decide how much to give. Let's model those two decisions separately, using:

$$E(\$|person) = P(response =' yes'|person) \cdot E(\$|response =' yes', person).$$

Note we need the first model to be well-calibrated because we really care about the number, not just the ranking. So we will try logistic regression for first half. For the second part, we train with special examples where there *are* donations.

Altogether this decomposed model makes a profit of $15,000. The decomposition made it easier for the model to pick up the signals. Note that with infinite data, all would have been good, and we wouldn't have needed to decompose. But you work with what you got.

Moreover, you are multiplying errors above, which could be a problem if you have a reason to believe that those errors are correlated.

Parting thoughts

We are not meant to understand data. Data are outside of our sensory systems and there are very few people who have a near-sensory connection to numbers. We are instead meant to understand language.

We are not mean to understand uncertainty: we have all kinds of biases that prevent this from happening and are well-documented.

Modeling people in the future is intrinsically harder than figuring out how to label things that have already happened.

Even so we do our best, and this is through careful data generation, careful consideration of what our problem is, making sure we model it with data close to how it will be used, making sure we are optimizing to what we actually desire, and doing our homework in learning which algorithms fit which tasks.