Rules of Machine Learning

(source: Martin Zinkevich, Google)

Some terms

- **Instance**: The thing about which you want to make a prediction. For example, the instance might be a web page that you want to classify as either "about cats" or "not about cats".
- **Label**: An answer for a prediction task either the answer produced by a machine learning system, or the right answer supplied in training data. For example, the label for a web page might be "about cats".
- **Feature**: A property of an instance used in a prediction task. For example, a web page might have a feature "contains the word 'cat'".
- **Feature Column:** A set of related features, such as the set of all possible countries in which users might live. An example may have one or more features present in a feature column. "Feature column" is Google-specific terminology. A feature column is referred to as a "namespace" in the VW system (at Yahoo/Microsoft), or a field.
- **Example:** An instance (with its features) and a label.
- **Model:** A statistical representation of a prediction task. You train a model on examples then use the model to make predictions.
- **Metric:** A number that you care about. May or may not be directly optimized.
- **Objective:** A metric that your algorithm is trying to optimize.
- **Pipeline:** The infrastructure surrounding a machine learning algorithm. Includes gathering the data from the front end, putting it into training data files, training one or more models, and exporting the models to production.
- **Click-through Rate:** The percentage of visitors to a web page who click a link in an ad.

Overview

- Engineering is more important than ML. If it's not reliable, solid, and reproducible, the rest doesn't matter.
- Have reasonable objects
- Be as simple as possible

Start without ML

- ML needs data, you rarely start with lots of data
- Heuristics will often get you half way there
- Your first goal is just to be better than random. So identify what random looks like.

Design and Implement Metrics

- Start by measuring, otherwise you can't know how you're doing
- Measuring the first thing is the hardest
- People care less early, so less resistance
- Get historical data now. When you start to care, you'll have a baseline.

Prefer ML to complex heuristics

- It's more maintainable
- But have you tried simple heuristics?

Start with simple models and get infrastructure right

- If your pipeline is shoddy, it will be hard to do anything anyway
- The first model provides the biggest delta
- This is "hello world" territory, focus on the basics
  - getting data
  - representing data
  - identifying good vs bad
  - how to integrate model into application
- Simple features are easier to understand, debug
- Make sure you understand your data

Test infrastructure separately from ML

- Make sure the infra is testable
- Make sure the ML is encapsulated
- Test getting data into the system
- Test that features are populated correctly
- Inspect the data (if allowed)
- Compare statistics from your pipeline with other sources (if exist)
- Test moving models from training to production
- Make sure you understand your data

Heuristics become features

- Often some system already exists. It uses heuristics, produces features. Take advantage of that.
- Consider using the existing system as a sort of pre-processor, generating synthetic features.

Monitoring and alerting are important

- Understand your freshness requirements
- Do sanity checks at model export time, at deploy time
- Understand what requires an email, what requires a page, what just has to be available for inspection
- Be aware of silent failures (e.g., data source decay)
- Make sure features have owners and that it's documented who they are (and that the features are documented). Same for algorithms.