Intro

This course is about applying Data Science to software engineering data. Data science is a broad term, but I will use it to describe using analytical techniques to support decision making. It combines hacking, statistics, domain expertise, and problem solving.

The analytical techniques can include:

- Descriptive stats like mean/median, variance, histograms, scatterplots
- Inferential stats like Bayesian inference, maximum likelihood, hypothesis testing
- Unsupervised clustering of data
- Predicting future values of data
- Finding a function that successfully captures the generative model e.g. with a neural network

Software Data

Like many areas, software development produces tons of data:

- Productivity stats like lines of code per hour;
- Communication patterns like developer code review histories;
- Natural language text like issues and bug report dicussions
- Code and code changes;
- Tool logs like build logs;
- Application specific metrics, such as uptime or fault reports.

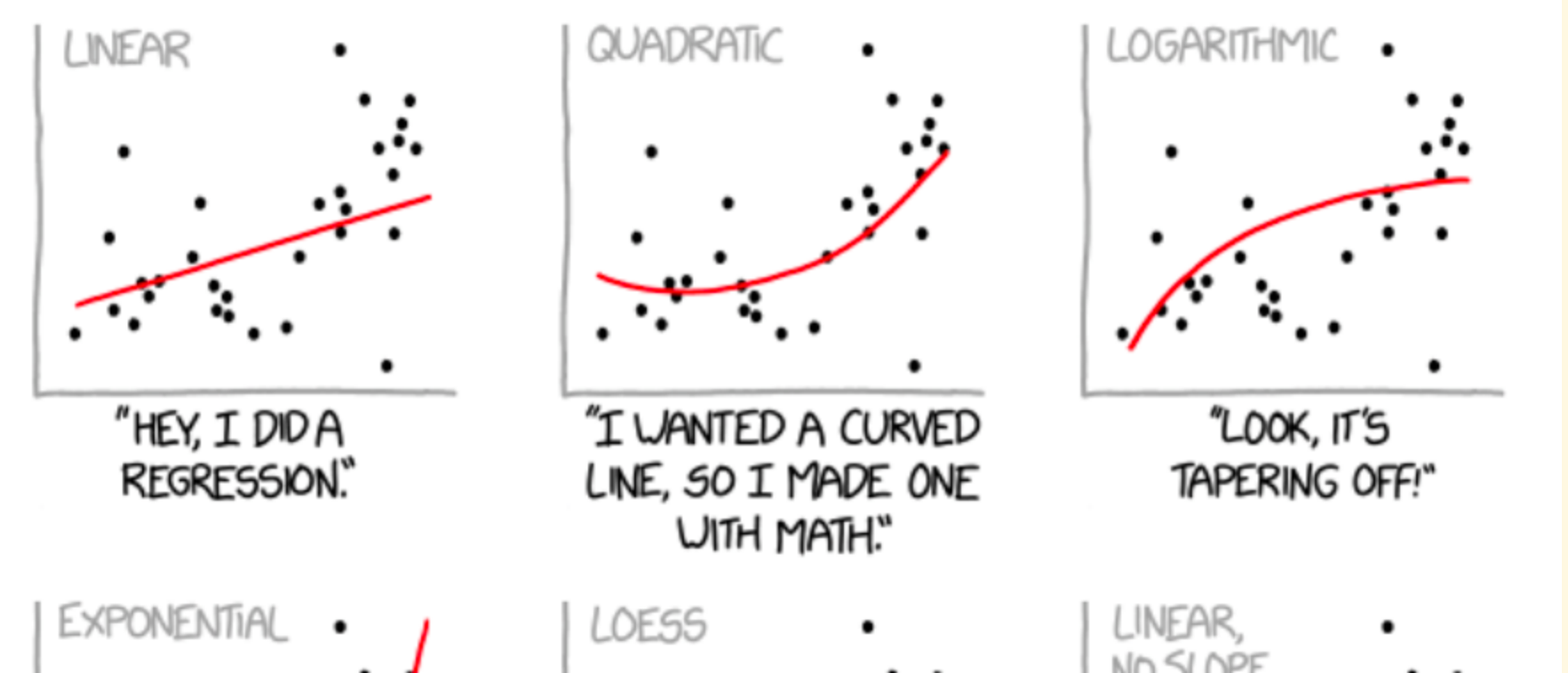
And many others. Note that some of the data is **nominal**, some **ordinal**, and some **interval/ratio** data. Some of it is **symbolic**, such as the structure of a program's Abstract Syntax Tree.

And some is **qualitative** data, which means it is not easily amenable to statistical analysis (often hiding the deeper issues).

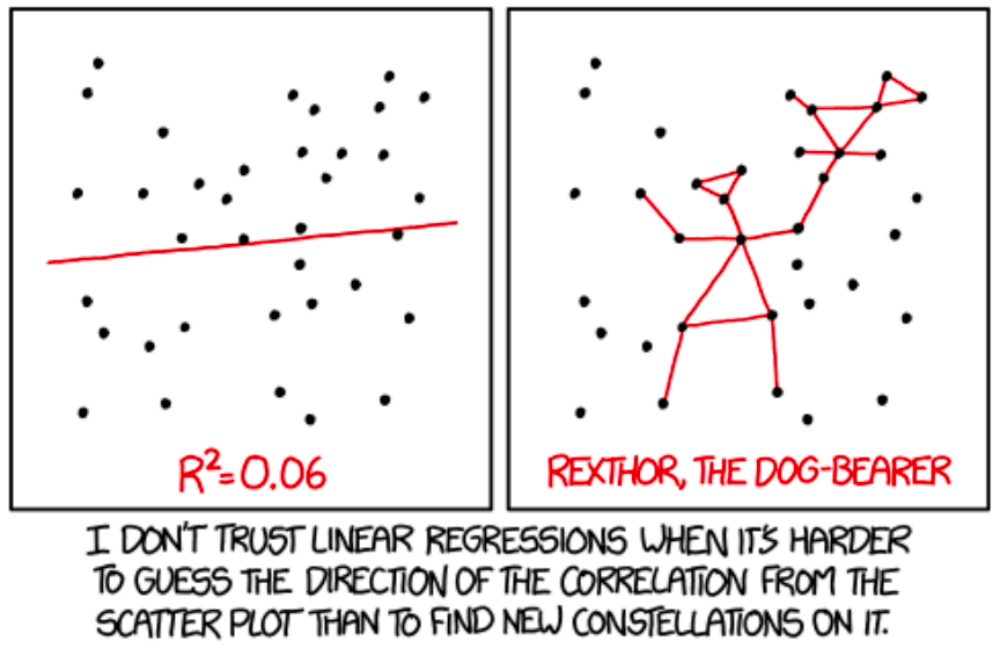
We can think about either learning directly from data or learning from models that generate data. In the Bayesian context, for example, what we will do is try to model the process that created the data we see, and learn the parameters of those functions.

For example, a lot of software data is log-normal, often zero-inflated, for example, the occurrence of bugs as we saw previously (many of the files may have no bugs recorded, and a rare few may have many many bugs).

“ The point of all the above algorithms is *choice*. Given some data (or model generating data) and some goals then there are many ways we use that data to achieve some, or all of those goals. To say that another way, we have many choices on how we put the world together: ”



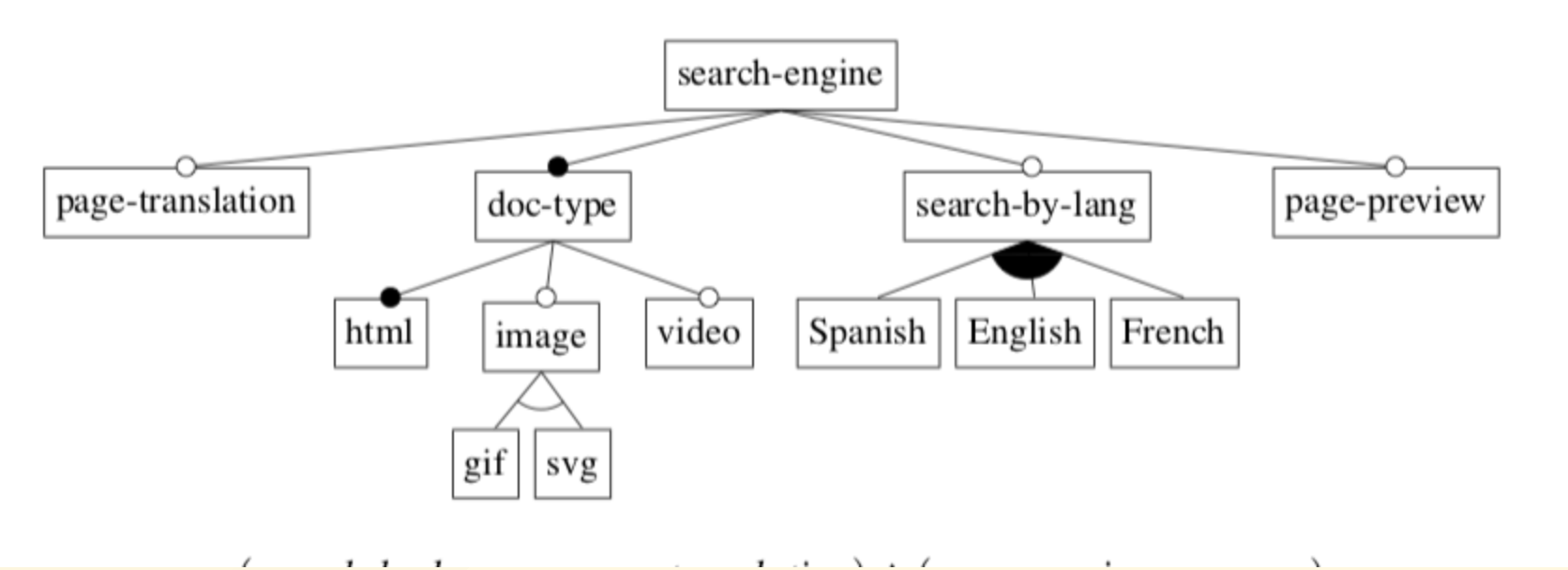
“ Some of those ways are quite silly: ”



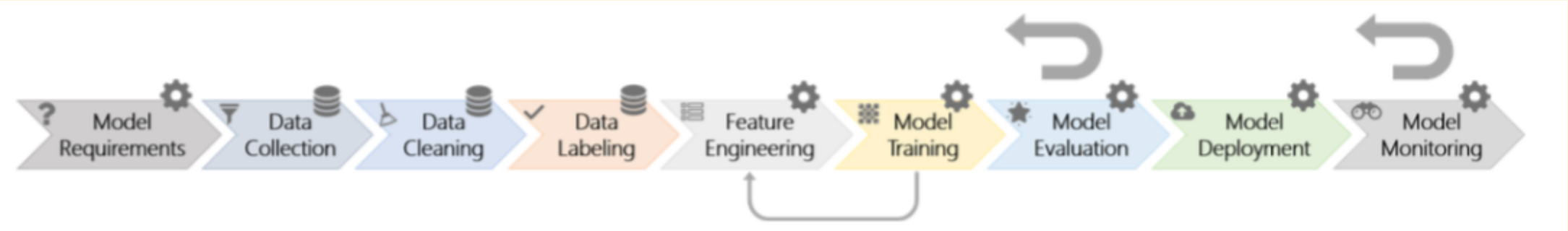
“ And some of those choices are very important.
For example Linares-Vásquez et al. use a genetic algorithm (which is a kind of optimizer) to propose better color schemes for cell phones. For example, of the screens shown below, the left-hand-side is an original, energy expensive design while the right-hand-side is far less energy demanding. ”


Original Design
CERCA PER NOME
AVVIA
COLORE FIORE
FIORITURA
TIPOLOGIA
25
GEMMA's lowest ECF solution
CERCA PER NOME
AVVIA
COLORE FIORE
FIORITURA
TIPOLOGIA
25

“ Mendonca et al. offer another example of exploring choice in software engineering. For example, software can be expressed as a set of choices. Here is a tree of options about a search engine. In this diagram, a filled/hollow circle means “mandatory”/”optional” (respectively). Also, white/dark fans means “and”,”or” (respectively). ”



Typically we follow this [nine step AI pipeline]:



Another way to think about this is using the *six steps of statistical modeling*:

1. specification (create a model)
2. identification (check if, given a new parameterization, your model's predictions change)
3. estimation (use the model to produce estimates)
4. evaluation (check the model; does the estimate match reality)
5. respecification (redo the model or try other models)
6. interpretation

Model comparison and exploratory data analysis

When presented with data or a theory about how data is created, what should we do?

- Explore the data with few preconceptions
  - look for the patterns
- Problem: this might bias us if the patterns are just noise

- Explore vs. confirm
  - Confirm: verify data support/reject hypothesis
- Hard to draw a line (Hullman and Gelman, 2021)
- Better intuition: explore means comparing data (typically visually) to a pseudo-statistical model (our prior).
- Then, create a more rigorous statistical model and compare alternatives.

And so we need to think about all of these stages in our production Data Science application: in this class we focus on the first 7. In general we will look at several types of tools:

- **Data miners**, that tell us what is in the data and build a model: nearest neighbors, decision trees, deep learners
- **Optimizers**, that tell us what to do, specifically, how to do something simple that has the biggest positive impact: genetic algorithms, heuristic search, etc.

And we can combine both: optimizing the results of a data miner, finding **optimal** sets of hyper parameters, etc. Using data mining to create models to optimize.

TPS exercise

think about the following question: "You are the software engineering manager or team lead at a big company. What is one key question you want to know the answer to for effectively running your team".

Write down a quick answer you have - maybe from your past experience.

Then *pair* with your table partner to *share* and discuss their answer and yours. We will then collect a few responses from the class as a whole.

Ethics

Since we can tweak data and analysis approaches in a variety of ways, one question we should think seriously about is ethical implications. People tend to ascribe a lot of weight to a data analysis, and ignore the many assumptions that underpin that analysis (e.g., how did we get the data in the first place? What was *Not* captured?)

Some ethical dimensions include privacy, regulatory compliance and safety, diversity and inclusion, etc. ...

Consider the two examples of advanced AI for code completion I attached to the readings, Github Copilot and OpenAI's Codex (both derived from GPT3). What are some possible issues with these tools?

Cross Tool Logs example

The reading this week looks at work by Google on understanding developer productivity (one dimension of it anyway).

The first thing to think about is a meta-analysis of the reading, and how to read academic research papers. Note how the paper is structured. This paper was published in a magazine, a more approachable format than others (journals or conferences for example).

The paper has a nice explanation of the motivation, and then gives a few short details on how it was done (why so scanty?).

The second aspect to think about is the methods used to produce insights. Were they all based on data mining?

The third aspect is the type of constructs the paper uses. One key construct here is the notion of *readability*. Do you agree with their definition?

The fourth aspect is to reflect on the results obtained. Do you believe the results? What might you do as a manager with these insights? What might be missing from the results?