Security data science – Getting the fundamentals right

Rich Harang
Director, Data Science

SOPHOS

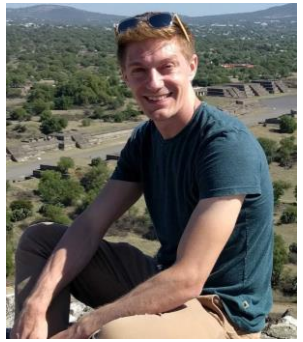
My background

- PhD in Statistics and Applied Probability (UCSB)
- Postdoc – UCSB Computational Science and Engineering Laboratory
- Five years at U.S. Army Research Laboratory
- Three years (and counting) at Sophos – ML for endpoint security
- Currently: Lead most “new research” within the Sophos Data Science team
  - Multiple projects, from one to five researchers working on any given one
- Most importantly:

I have stepped on almost every rake there is, so you don’t have to.

Data Science @ Sophos

Josh Saxe



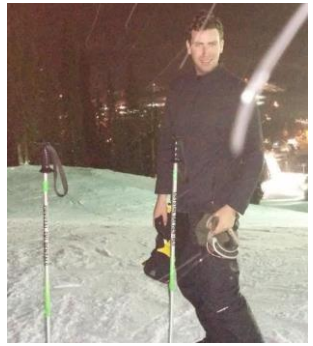
Kevin Hake

Rich Harang

Hillary Sanders

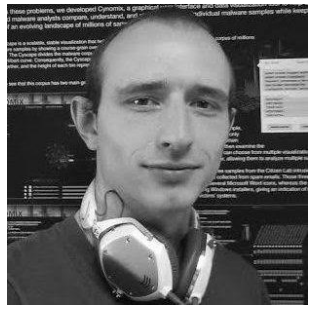
Adarsh Kyadige

Konstantin Berlin



Ethan Rudd

Felipe Ducau



Alex Long

Andrew Davis

William Lee

Matt Stec

Matt Burnett

You?

About this talk

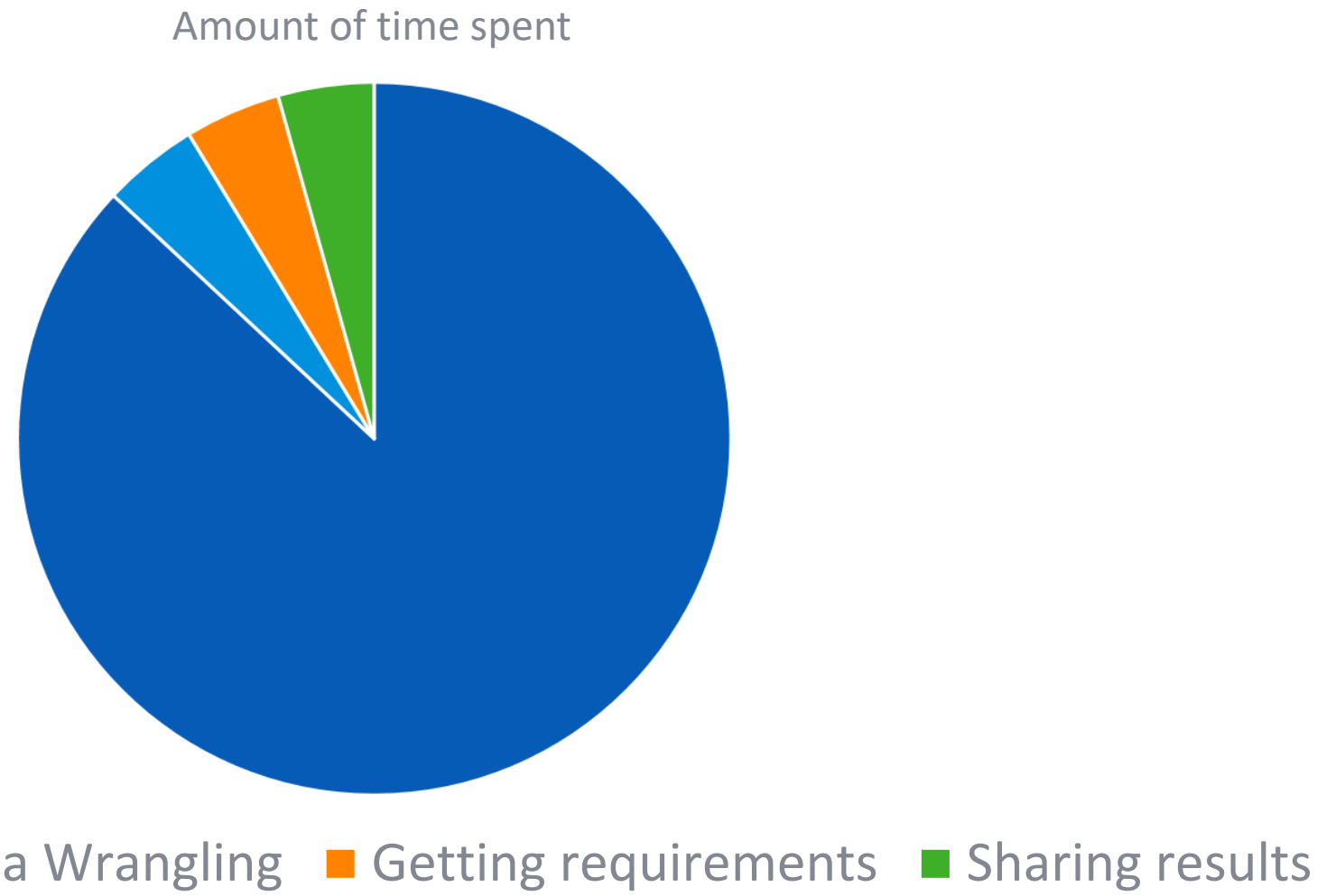
- What I will talk about: What/how/why of security data science
  - **What?** A model? A continuous analysis? A single decision?
  - **How?** Best practices, making the process go as smoothly as possible
  - **Why?** Making sure that what you make is what your organization needs

- What I'm not going to talk about:
  - Specific security problems, kinds of models, frameworks, or the latest academic research

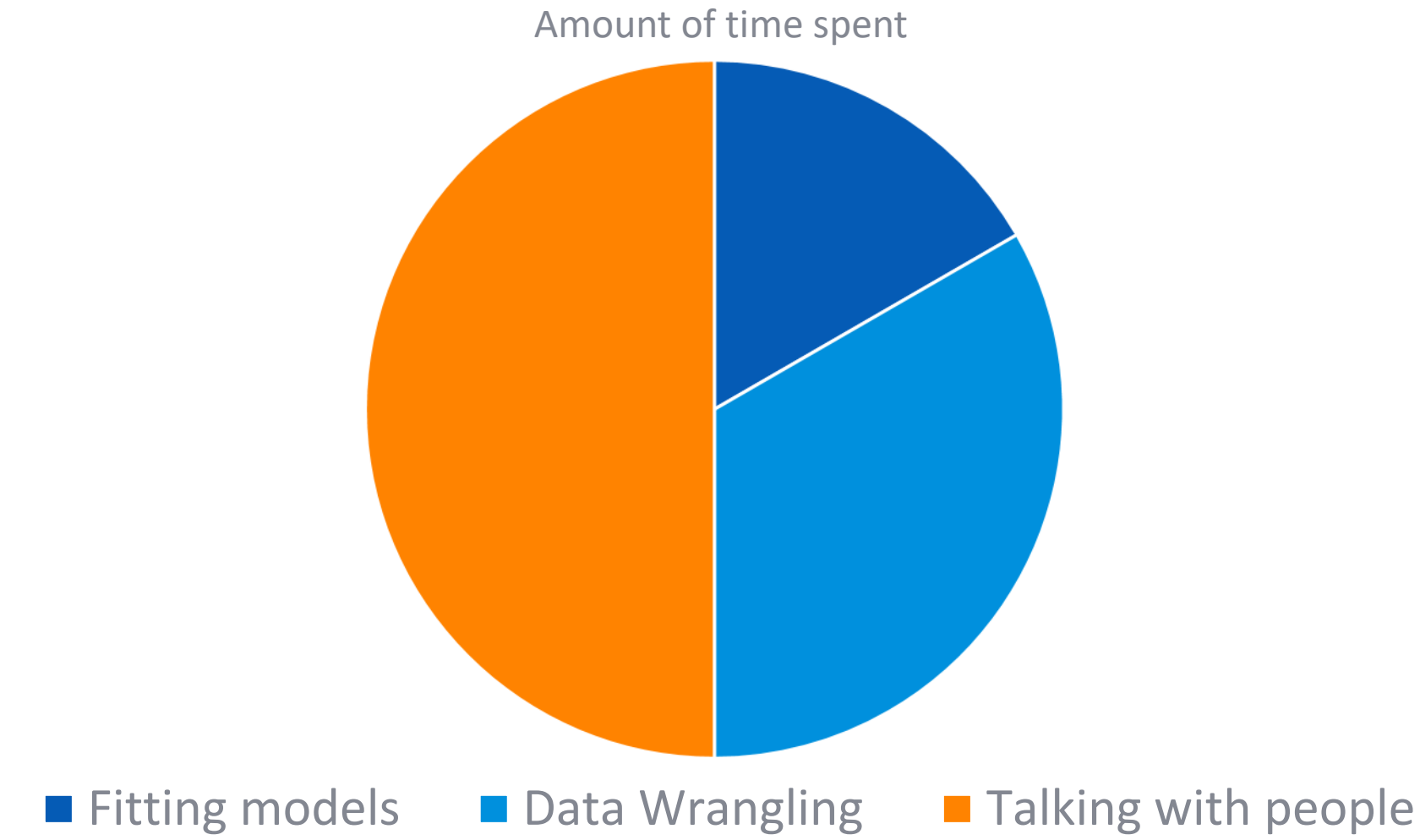
If you remember nothing else from this talk…

SOPHOS

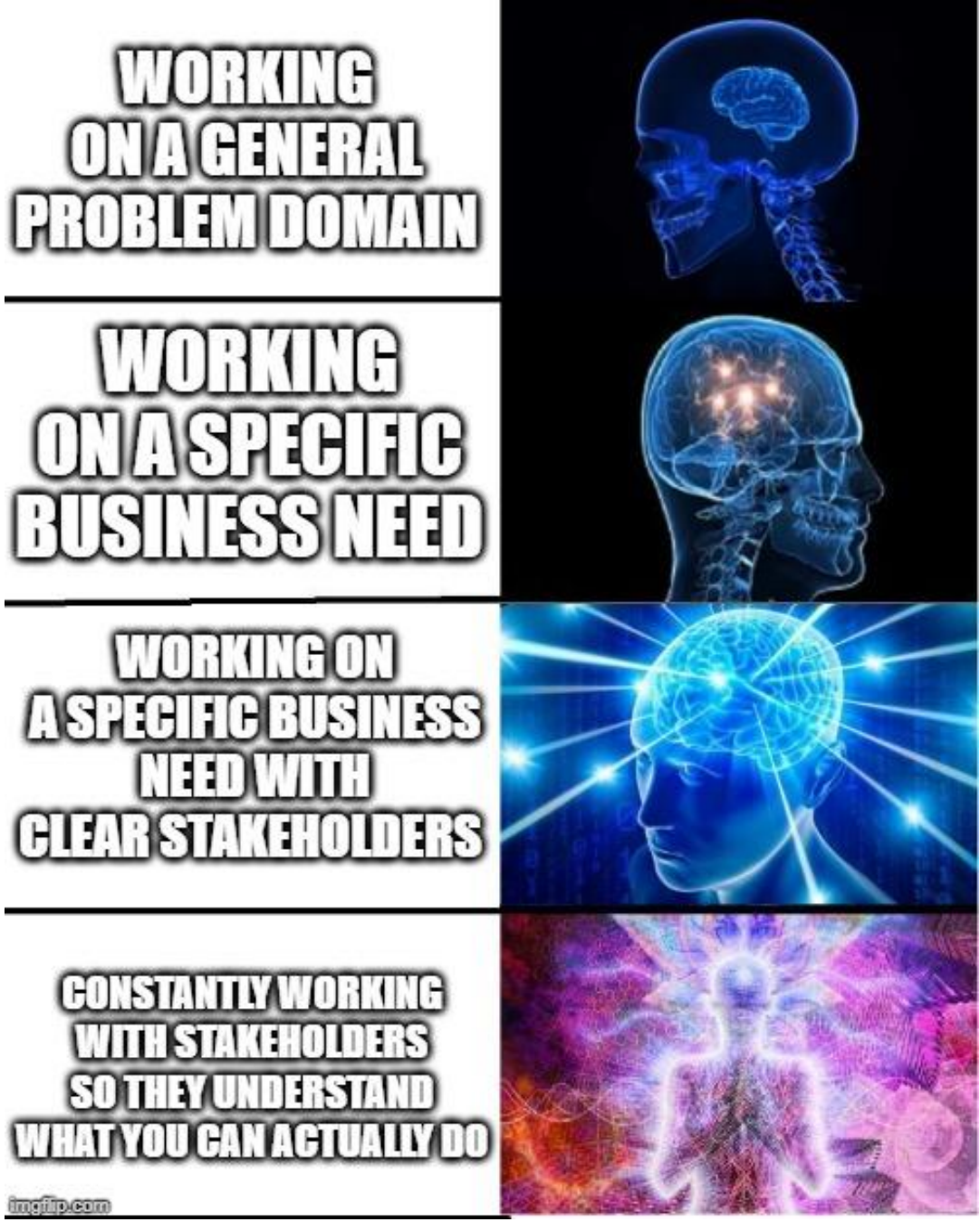
What you think your job as a data scientist is



What it actually is



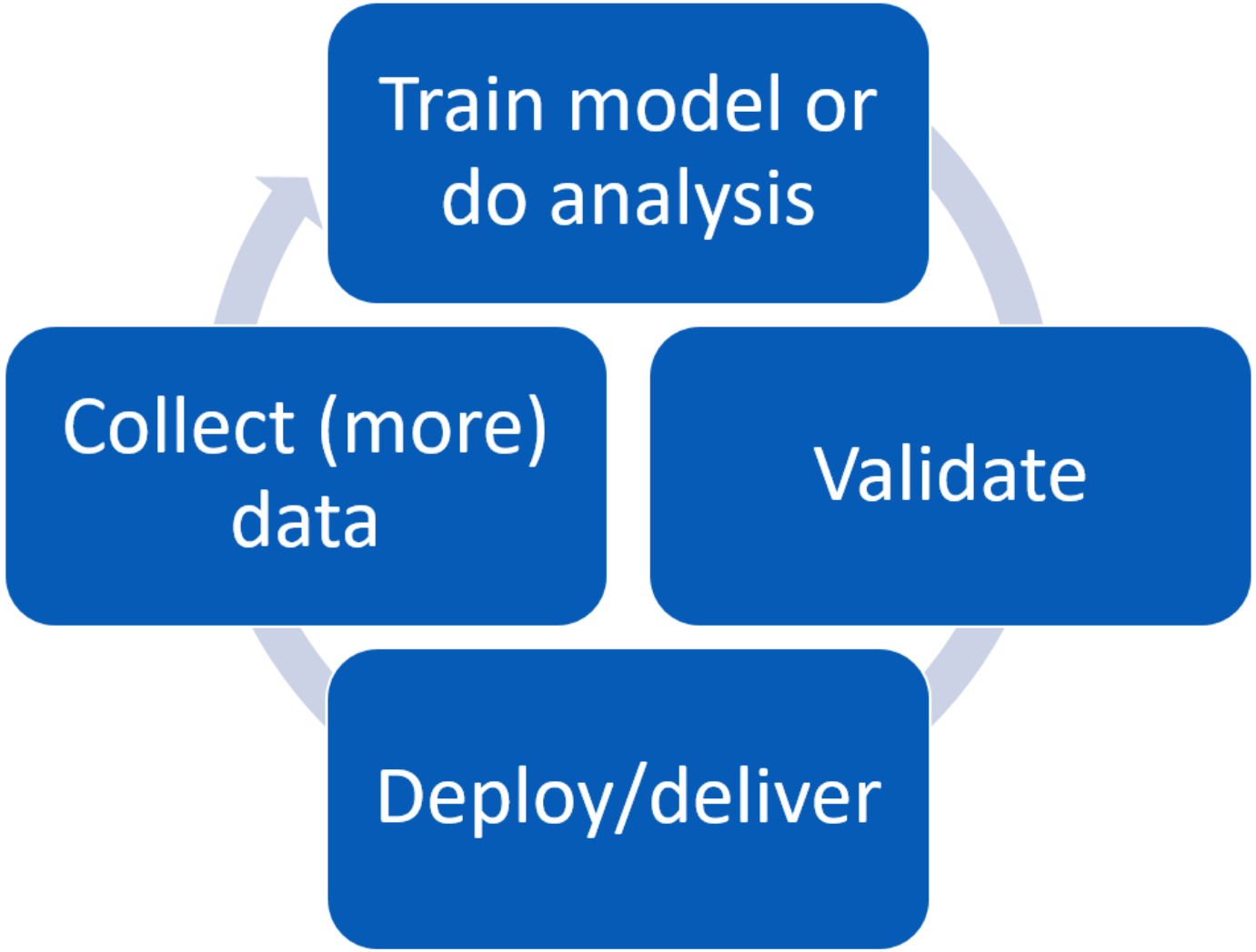
The single best predictor of success for a security data science project is communication and collaboration.



The “Why”

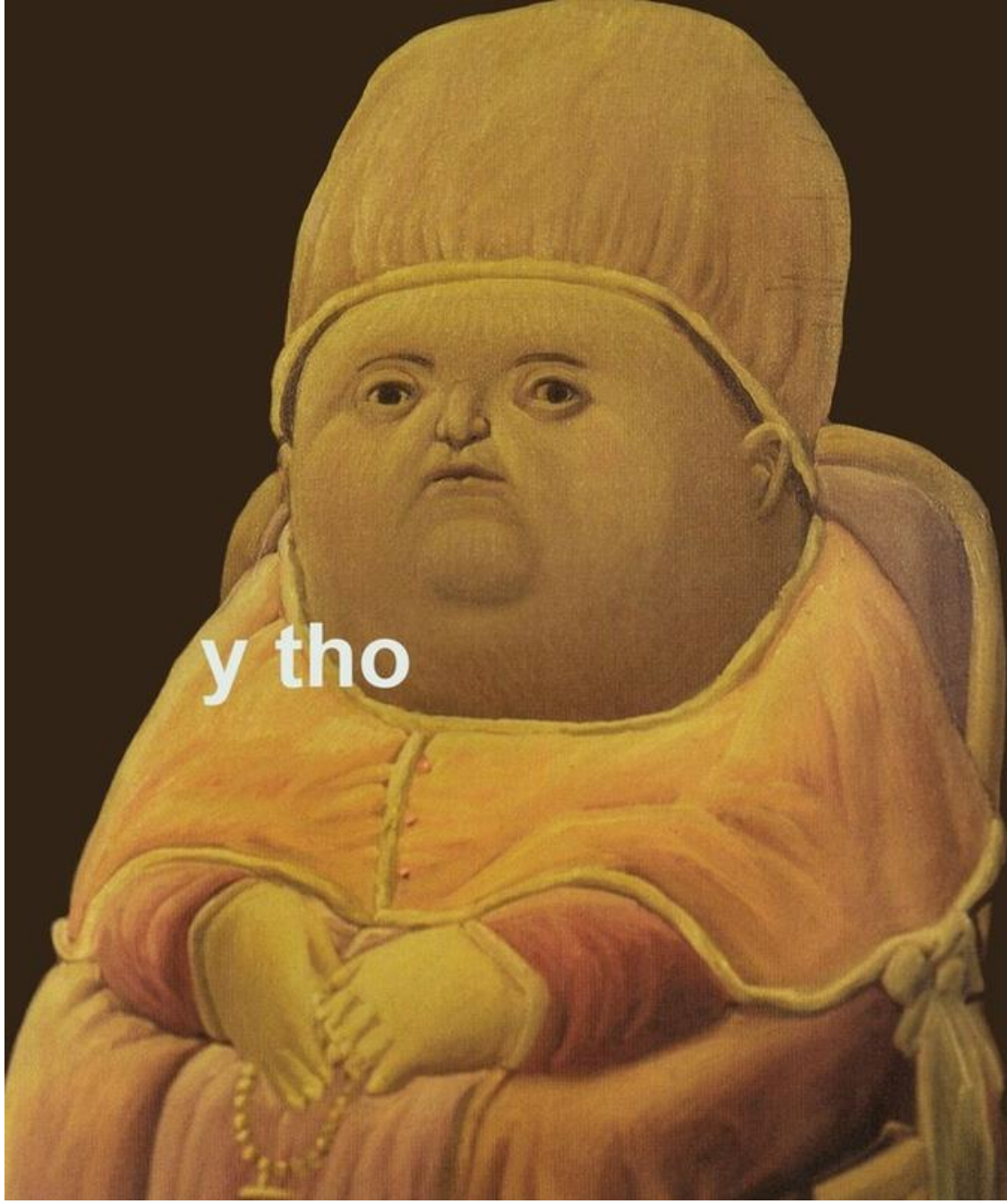
SOPHOS

The Data Science process in theory

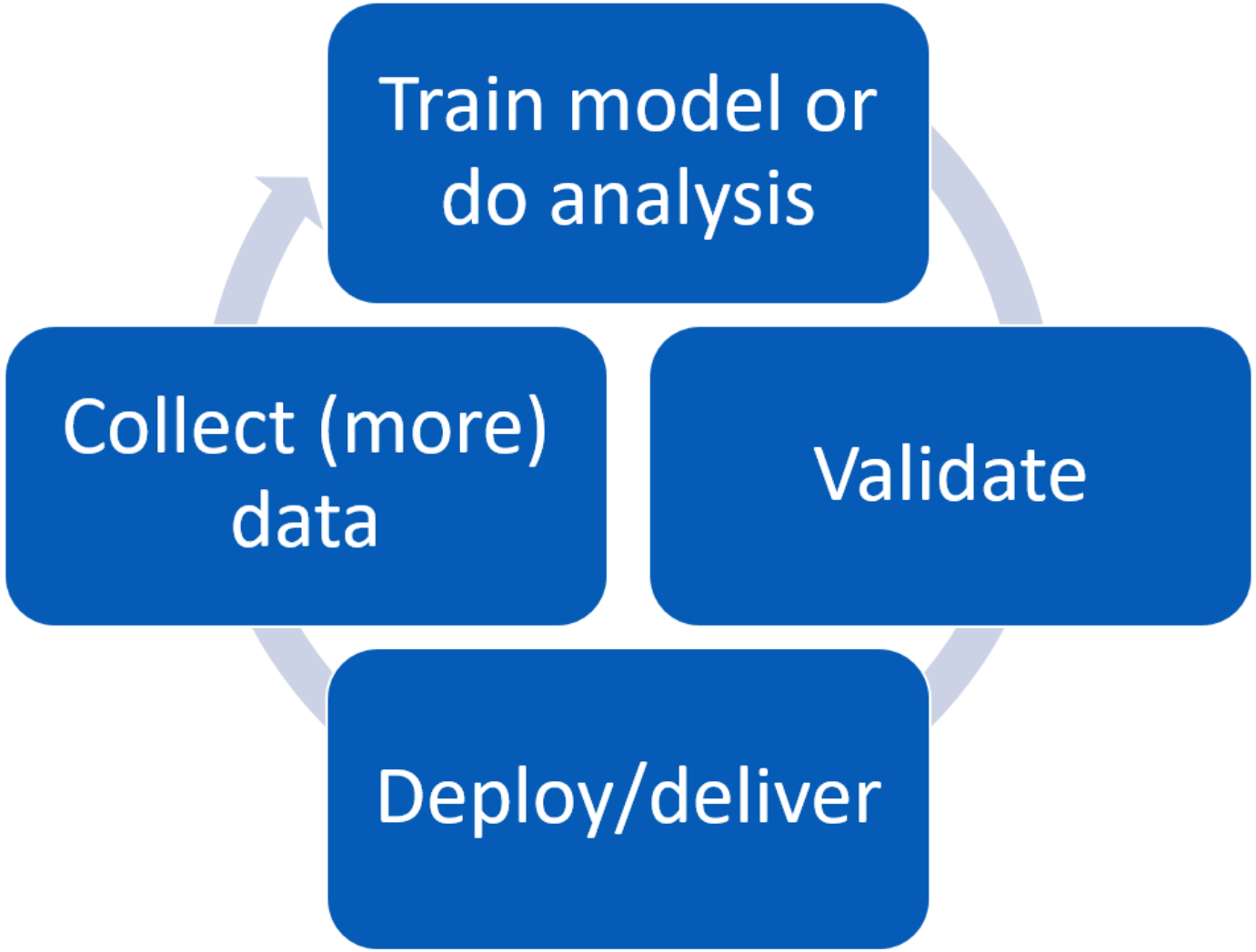


The very first question:

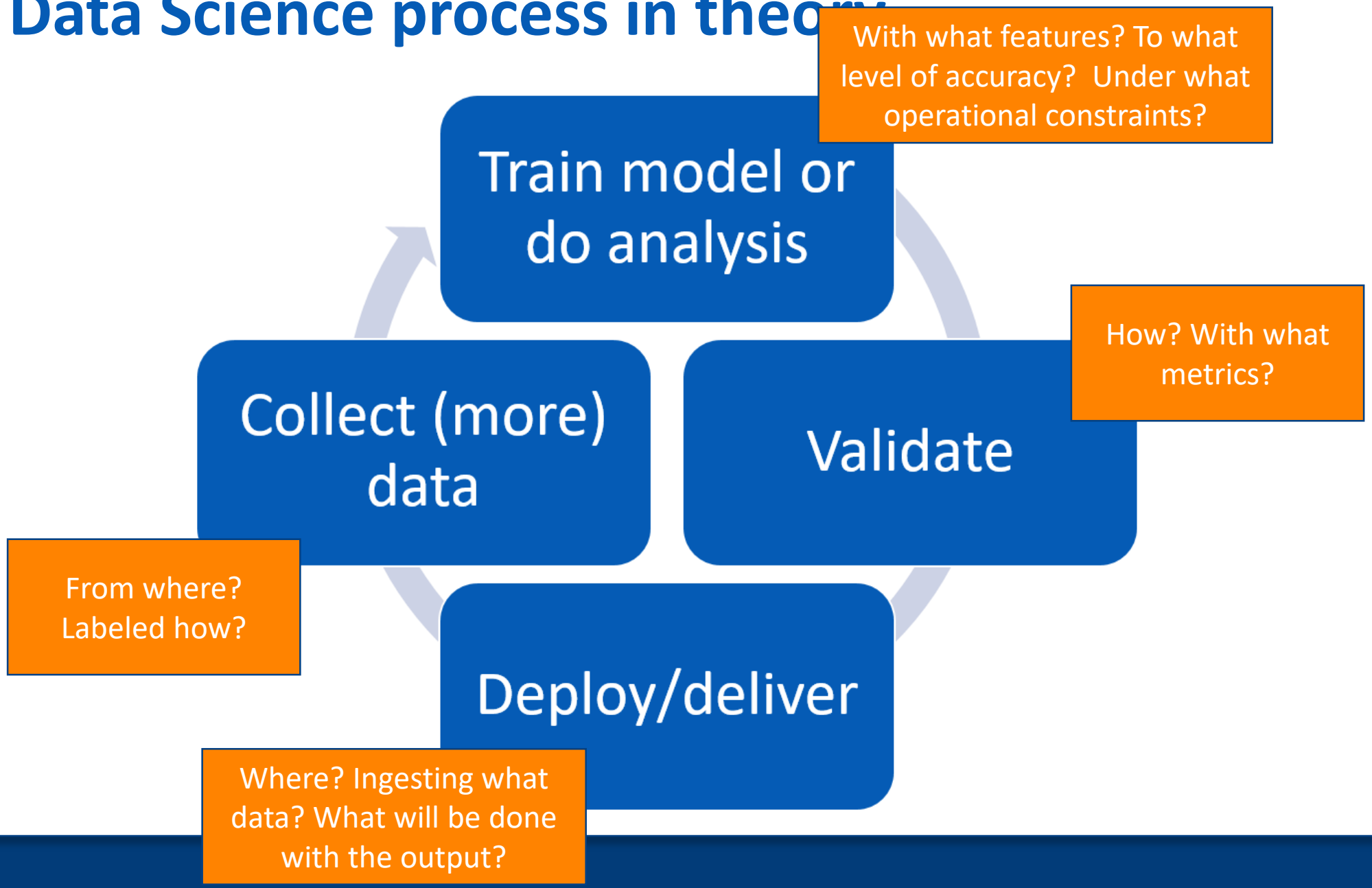
- Why are we doing this?
  - How will this model or analysis be used?
  - Can we just answer a single question and then stop?
  - What is “good enough”?
  - **What do we actually hope to accomplish here?**



The Data Science process in theory



The Data Science process in theory



They may not know what they want

- You have to be an educator – What are the tradeoffs you can offer?
- What is realistic to expect from the analysis or model in that problem space?
- Can you map their goals for the project to something concrete and measurable?

Before the “How” part begins…

YOU should know how your project will fit into the larger business process.

THEY should understand what you’re giving them and how to interpret or use it.

EVERYONE should agree on what success looks like and how to measure it.

The “How”

SOPHOS

The “How” of a good project

1. Get good data; clean it, curate it, protect it
2. Have good practices for internal collaboration
3. Construct good features, track them, and evaluate their usefulness
4. Be scientific: use common metrics, be able to link results to the code that produced them, share results regularly
5. Transition/deploy intelligently: agree on interfaces, know your constraints
6. Track and monitor: get feedback from the real world

All about data

SOPHOS

“The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted.”

– John Tukey

"If you torture the data long enough, it will confess."

– Ronald Coase

Know your data

“The best material model of a cat is another, or preferably the same, cat.” – Norbert Weiner

- Look at the real world data your project will consume as early as possible

Things to consider:

- Label distribution
- Label bias
- Data collection bias

The Security Difference

- “The enemy gets a vote”
- Labeling can be hard – they’re trying to hide
- Data distributions change over time as enemies adapt
- It might not be worth using old data!

So what should you do?

- Data sources
  - Can you get telemetry from deployment?
  - Are other people working on similar problems? Will they share?
  - Can you buy the data outright?
  - Is there an existing dataset (e.g. kaggle competition) that could be repurposed?
- Labels – think carefully about these!
  - “Vendor aggregation” sites and feeds – voting over detections is a common choice
  - Crowdsourcing sites (e.g. Amazon Mechanical Turk, other more specialized services)
  - Self-supervised labeling from a ‘seed corpus’
- Train/test splits: mimic deployment
  - In most cases, this means a single temporal split: train on the earliest data, validate on the middle chunk, and test on the newest data

Be aware of risk/regulatory issues (GDPR, breach risk, etc.)

Security data isn't oil, it's plutonium: you want the highest quality you can get, but it can be dangerous to hang on to and it's got a half-life.

- Talk with Legal early and often
- Think very carefully about what you actually need to store
- Think very carefully about how long it's worth storing it
- Know the risks posed by losing control of it or accidentally leaking it
- Consider transforming the data into a less risky form or subset
- **Talk with Legal early and often**

Finally

Once you have a good data set, *protect it*:

“raw data” should be read-only

Internal collaboration

SOPHOS

Have one way of doing the important things

- Agree on a single set of tools, frameworks, etc.
- Avoid “not invented here” syndrome
- Clearly separate utility code from experiment- or analysis-specific code
- Treat in-house utility code as the mission-critical software engineering project that it is

Metrics code

Everyone needs to use the same metrics code:
NO EXCEPTIONS

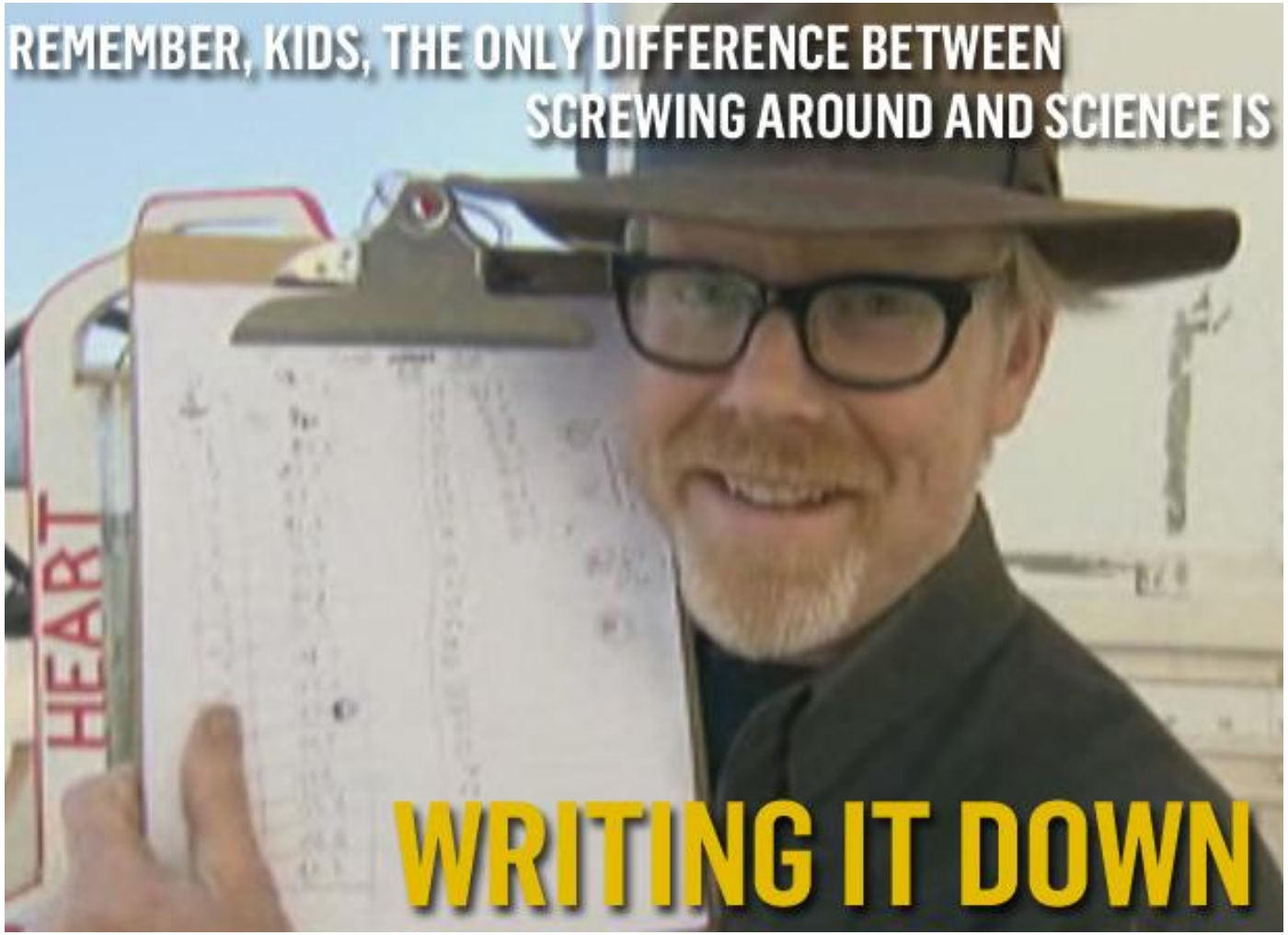
Metrics code

Everyone needs to use the same metrics code:
NO EXCEPTIONS

NO EXCEPTIONS.

Documentation – the only way to collaborate across time

- Documentation is a continuous process
- Well-documented code should always be part of the deliverable
- Even for ML models, plan on writing an after-action review; keep some form of “lab notebook”



A non-exhaustive list of questions to answer before you start

- What is the common data set that is everyone going to use? How will it be split?
- What are the right metrics? Which specific functions will be used?
- What format are you going to store results from the experiment in? Where will those results live?
- Where does documentation live, and what is ‘mandatory’ to store in it?
- Where will the source code live, and how will it be managed?

The care and feeding of features

SOPHOS

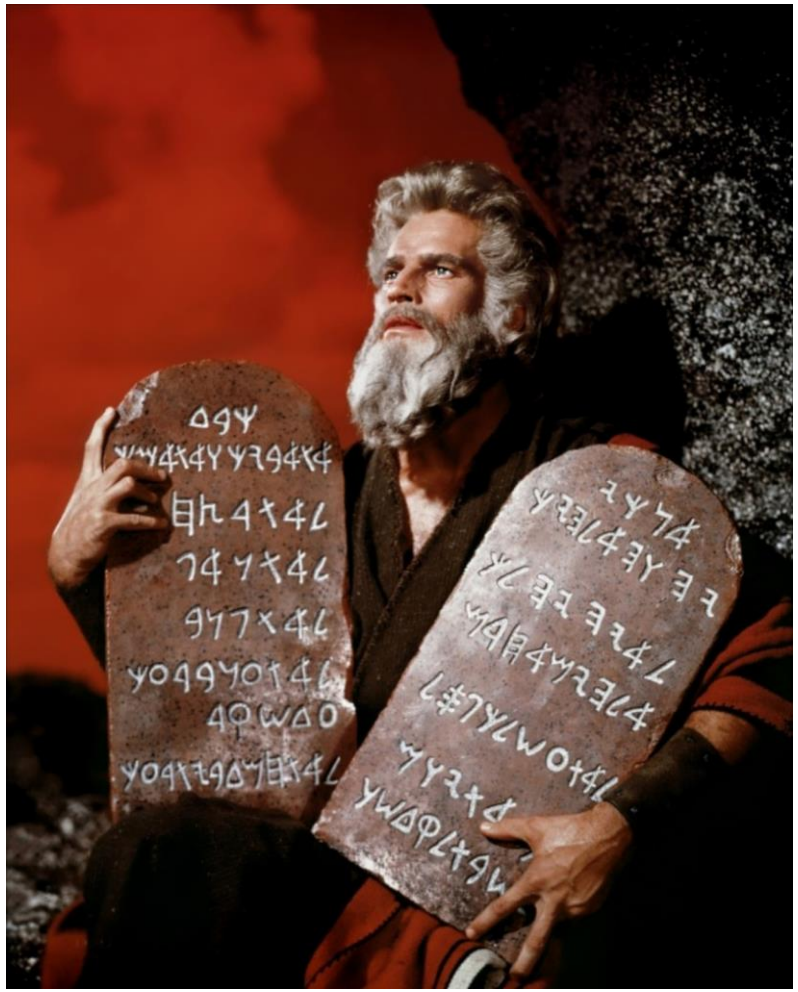
“None of the models I’ve tried seem work.”

“I was getting awesome results, but something changed in my feature extraction code and I don’t know what and now everything’s terrible.”

“It worked great during research but it’s hot garbage in production.”

The ~~ten~~five commandments

- You shall **always** talk to domain experts and heed well their advice
- You shall **never** run feature extraction unless it’s checked in to version control
- You shall **always** be able to link a set of features to the commit that produced them
- You shall **always** be able to link an experiment to the commit of any features used in that experiment
- You shall have (at a minimum) **test vectors** for feature extraction functions

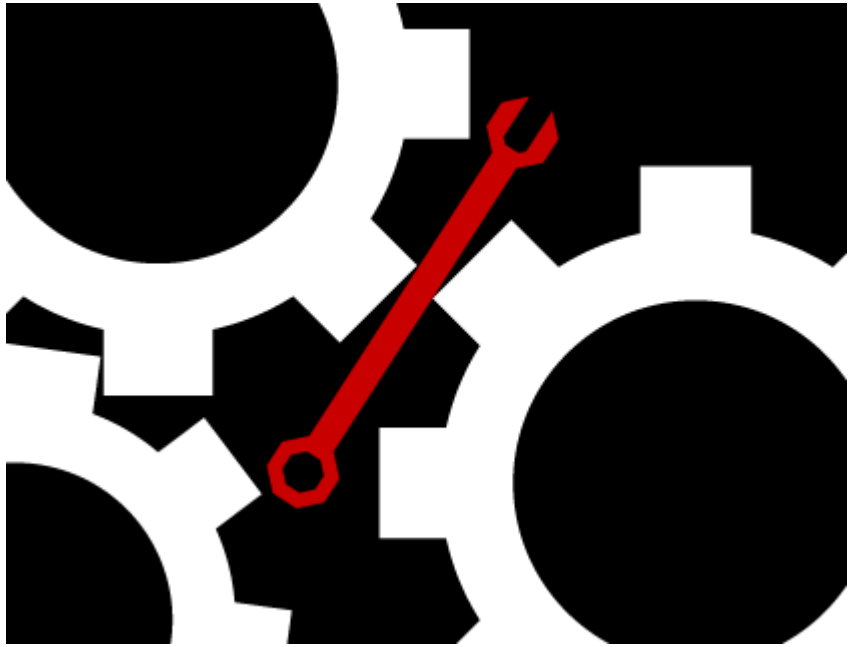


Domain experts are, well, experts

- What do they look at first?
  - Get past the “Just don’t look right” stage
- What tools do they use?
  - Can they be automated for feature extraction?
- How do the bad guys try to hide from them?
  - Does that suggest features?
- How would they break or evade your favorite feature?
  - Start studying robustness before the bad guys do

The security difference – antiforensics and parsing headaches

- Anti-reversing and obfuscation techniques are commonly used, but may be rare within your data
  - Can fatally break feature extraction
  - Make sure it can recover gracefully
  - Keep track of which samples break your feature extraction
- What will you do if some or all features can't be extracted?
- Constantly revisit features in light of new attacks and evasions



Put the “Science” in “Data Science”

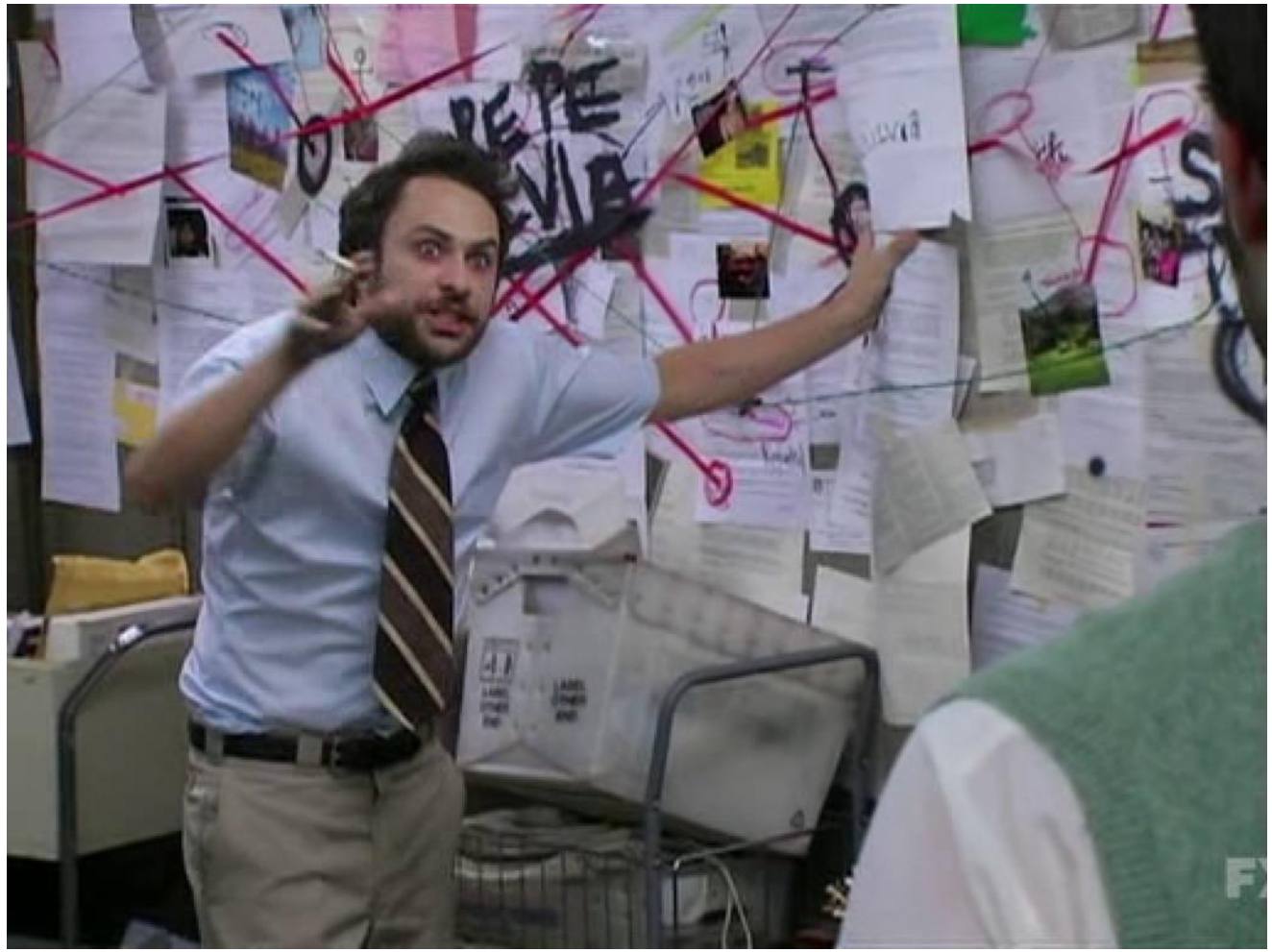
SOPHOS

The zeroeth step: use Occam's Razor

- **Don't** reach immediately for the most complicated model or technique in your toolbox

- Establish good baselines so you know what you gain by adding complexity

- Always start with the classics:
  - Linear model
  - Naïve Bayes
  - Random forest
  - Etc.



Key questions

- Can anyone on this project…
  - …directly and meaningfully compare the results of any two runs?
  - …immediately reproduce the results of any other person?
- If one set of results looks better, can we easily identify why?
- Six months from now, will someone else be able to figure out…
  - …why you pursued one line of research and ignored another?
  - …what exact command line you used for your best model or analysis?

Key questions

- Can anyone on this project…
  - …directly and meaningfully compare the results of any two runs?
  - …immediately reproduce the results of any other person?
- If one set of results looks better, can we easily identify why?
- Six months from now, will someone else be able to figure out…
  - …why you pursued one line of research and ignored another?
  - …what exact command line you used for your best model or analysis?

Are you sure?

Manage your experiments

- Aim for simple commands like `python experiment.py`
  - Avoid command-line parameters
- Saving experimental artifacts
  - Log files, results, models, plots
- Save experimental parameters
  - Not on the command line
- Save or track critical source code
  - E.g. git commit shorthash
- Store dependency information
  - E.g. via dumping `conda env`
- Save/require notes
  - “I’m doing this run to examine X”

Know when to shift gears

1. “Tinkering” – does this idea have legs? Is it worth exploring further? What plots or visualizations seem useful and/or interesting?
   - REPL/Jupyter notebook work.
2. “Testing” – The idea seems valid, can it be scaled? Does it continue to work on larger data sets? What variants are worth full scale testing?
   - Experimental scripts; basic logging
3. “Rigorous study” – Full scale test, hyperparameter search etc.
   - Clean code with separated functionality in a good framework

Know when to shift gears

1. “Tinkering” – does this idea have legs? Is it worth exploring further? What plots or visualizations seem useful and/or interesting?
   - REPL/Jupyter notebook work.
2. “Testing” – The idea seems valid, can it be scaled? Does it continue to work on larger data sets? What variants are worth full scale testing?
   - Experimental scripts; basic logging
3. “Rigorous study” – Full scale test, hyperparameter search etc.
   - Clean code with separated functionality in a good framework

You cannot do a large, collaborative data science project with only Jupyter notebooks; go ahead and @ me.

Do your experiments support your conclusions?

- Change one thing at a time
- Re-run leading candidate experiments or analyses to ensure reproducibility/stability
- Criticize your own work: how could Nature and Chance have conspired to lead you to the wrong conclusion?

Does your data support your conclusions?

- Not all metrics are meaningful
- Not all measurements are precise
- You might not have enough data to measure what you want to measure

The security difference – noise and drift

- Your ground truth labels probably aren’t perfect
  - Are you over- or under-estimating error rates due to systematic error in your ground truth labels?
  - Are you reporting statistics below the noise floor?

- Drift (adversarial or otherwise) is always a concern
  - Do you know how fast your data distribution changes over time?
  - Could old/outdated data be biasing your measurements?
  - Are you changing the data distribution by measuring it?

Deploying without YOLOing

SOPHOS

You *have* been talking with the customer regularly, right?

- You and your customer should both be on the same page:
  - what the model/analysis covers
  - where to plug it in, inputs/outputs
  - any potential issues or caveats with the model or the results

- If needed, you should be able to explain any modeling or analysis decision you made on the basis of existing documentation
  - Also – samples you couldn't analyze, feature issues, etc.

Good. So here's what's left:

- Give them a mockup to smoke test with.
- Do whatever your pre-agreed conversion from "research" to "production" is.
- **Use your test vectors:**
  - Do your production features match your research features?
  - Do your production feature extractors fail gracefully on 'bad' input samples?
  - Does your production model give the same results on those features as your research model?
- Package and begin the handoff.

Easy, right?

- In practice, this is *incredibly hard*, especially the first time
- Talk with your customers, be flexible about how you deploy
- Get it into a standardized, repeatable with a stable interface process as quickly as possible

The Security Difference – bridging the culture gap

Data Science deals in probability and aggregate statistics, not individual artifacts

- Don’t declare defeat because you missed a particular event or instance
- Don’t declare victory because you caught a particular event or instance
- **Expect individual errors, have a plan to handle them**
- Don’t try to move from population-level statistics to conclusions about individual items – it’s a census, not a case study

Again: education and careful communication is important

“So, uh, did it work?”
Tracking and telemetry

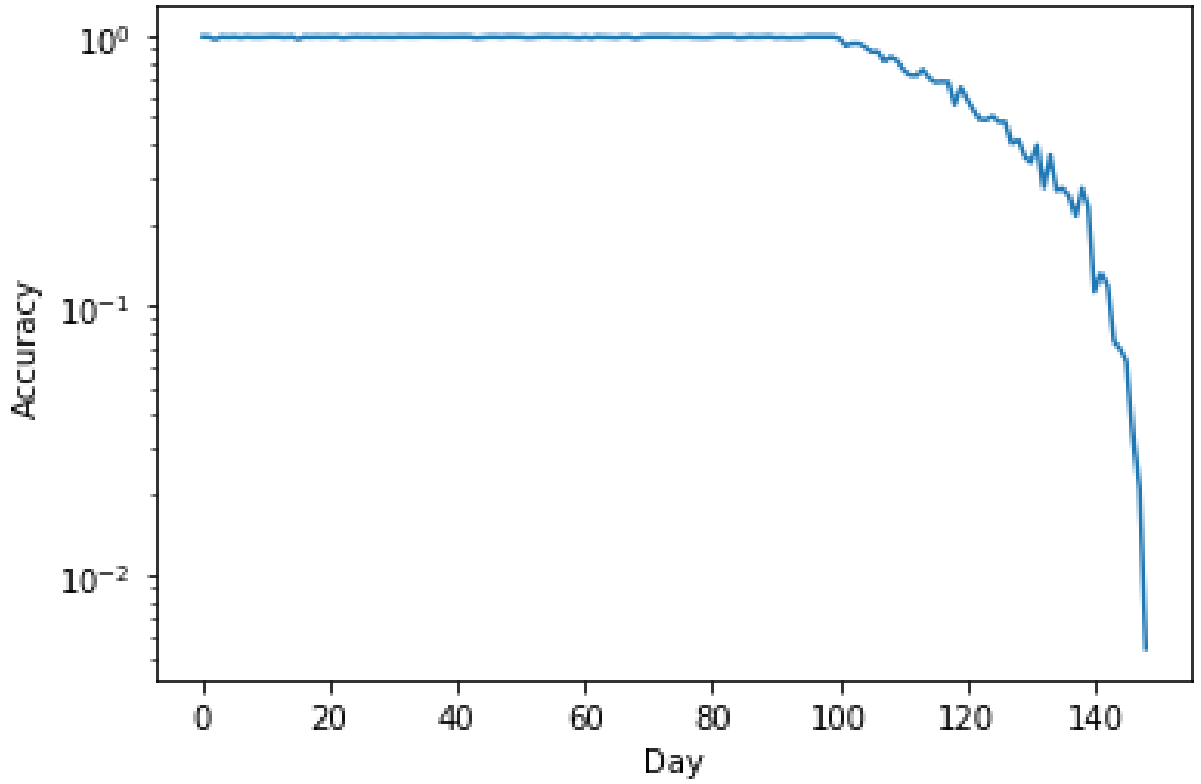
SOPHOS

You are getting telemetry, right?

- Is the model performance on “real” data close to what you expected?
  - Consider a “silent” deployment at first
- Does the “real data” look like what you expected?
- Is the model adding value?
- Can you use the model to collect real-world data for later refinement/training?
  - Beware data storage issues though! PII? Retention? Breach risk?
  - Talk to Legal early and often!

The Security Difference – Adversarial model decay

- It'll work great until it doesn't
- Always be cross-checking model performance against new threats, attacks, and campaigns
- When something new is identified, go back through your data and see how long it's been around and what your model did with it



In conclusion

SOPHOS

What did we learn from all this?

This:

1. Deliver a Data Science project that someone wants
2. Find out what they want by talking to them
3. Do good science
4. Make sure they know what you've delivered and how to use it
5. Keep an eye on what you delivered and be able to tell when it's not working anymore

Build the systems that enable all of the above.

Questions?

SOPHOS

SOPHOS

Cybersecurity evolved.