# DevOps for machine learning and other half-truths:

## Processes and tools for the ML lifecycle

Diego Oppenheimer, EVP MLOps



DataRobot

# Speaker Bio

Diego Oppenheimer is co-founder and CEO of Algorithmia, the enterprise MLOps platform, where he helps organizations scale and achieve their full potential through machine learning.  Algorithmia puts ML models into production faster and more cost-effectively with enterprise-grade security and governance. Diego is active in AI/ML communities and works with leaders to define ML industry standards and best practices. He brings his passion for data from his time at Microsoft where he shipped some of Microsoft's most used data analysis products including Excel, Power Pivot, SQL Server, and Power BI. Diego holds a Bachelor's degree in Information Systems and a Masters degree in Business Intelligence and Data Analytics from Carnegie Mellon University.

**Diego Oppenheimer | EVP, MLOps**

**@doppenhe | linkedin.com/in/doppenheimer**

# How do I deploy this model?

# Machine Learning != Production Machine Learning

Distributed parallel processing

Load balancing

Cloud infrastructure decisions

Cluster orchestration

Model versioning

API management

Container image management

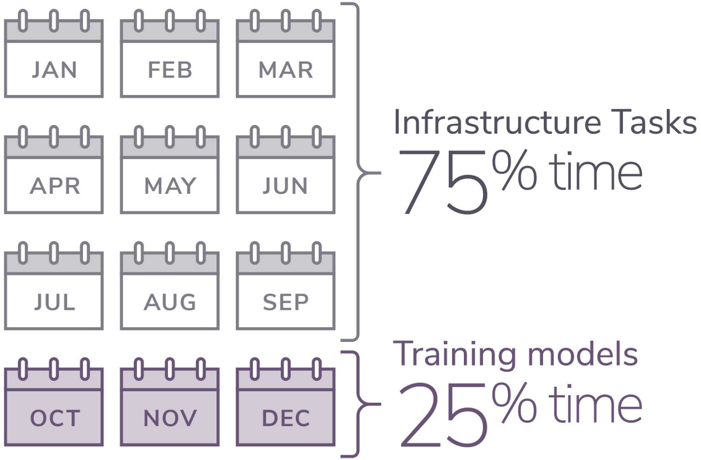
Utilizing GPUs

**DevOps:** "a set of practices intended to **reduce the time** between committing a change to a system and the change being placed into normal **production**, while ensuring high quality"

# Survey: Teams are capable of much more

**75% of time spent on infrastructure**




**Key challenges**

**30%:** supporting different languages and frameworks

**30%:** model management tasks such as versioning and reproducibility

**38%:** deploying models at necessary scale

* - survey of > 500 practitioners & management in summer of 2018

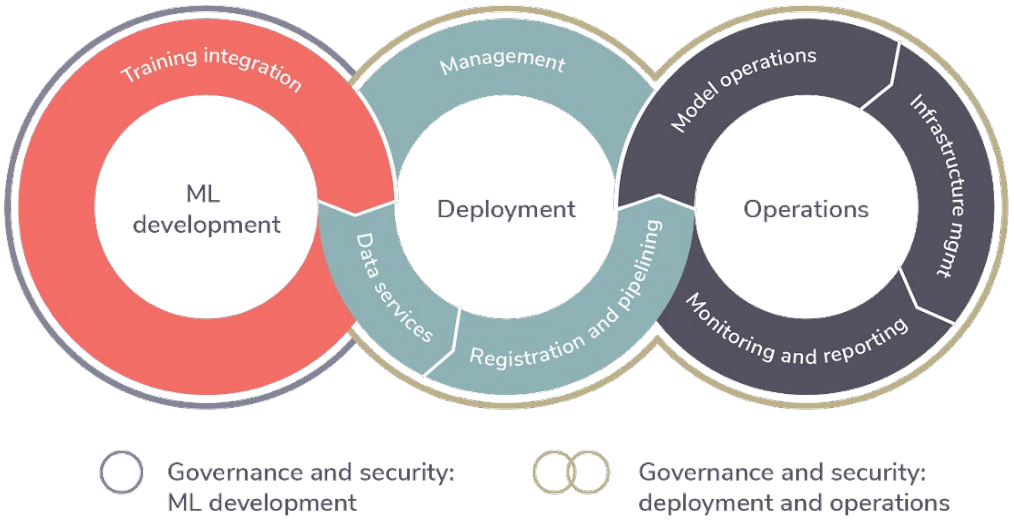
# Traditional Software : Machine Learning
# SDLC : MLLC

**Software dev lifecycle**

| Git | Jenkins, GitLab | AWS, Heroku, Kube | ELK, Grafana |
| --- | --- | --- | --- |
| ↑ | ↑ | ↑ | ↑ |
| Source control | CI | **CD** | **Manage** |

| | Source control | CI | CD | Manage |
|---|---|---|---|---|
| **Software dev lifecycle** | Git | Jenkins, GitLab | AWS, Heroku, Kube | ELK, Grafana |
| **Machine learning lifecycle** | Git | Test set | Needs: GPU, Cold Starts, etc. | Continuous evaluation |

# How is ML different?

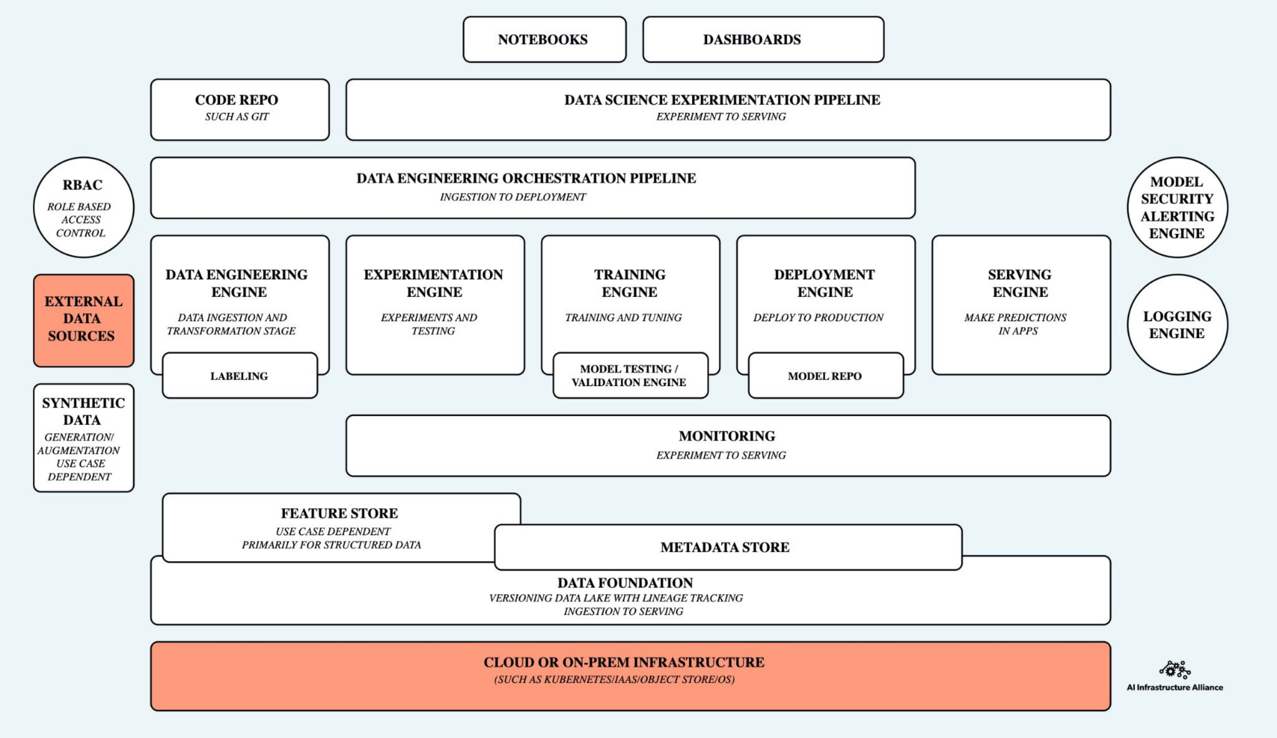



- ML development lifecycle is an evolving ecosystem
- ML moves faster than traditional app dev
- “Upgrades” in ML often come from more data, not new code

# Boyd’s Law of Iteration

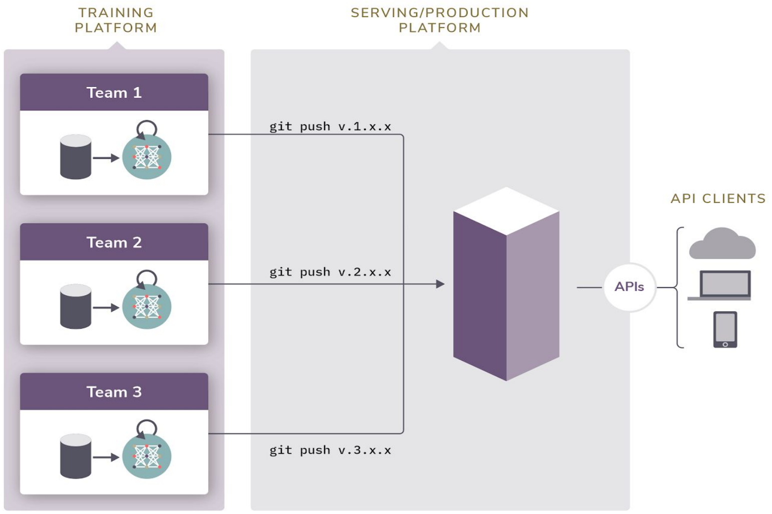
Data — Train — Deploy — Manage



Data
Train
Deploy
Manage

# Data is useless without algorithms, but algorithms are also useless without data.




## Training and production are very different

### Training

- Long compute cycle
- Interactive
- Exploratory
- Stateful
- Single user

### Production

- Short compute bursts
- Elastic
- Stateless
- Many users

**How to bridge the gap between training and production?**

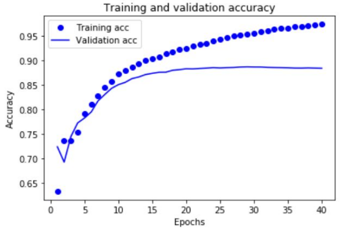
**Proposed solution: The 2-Git Flow**

- Machine learning, unlike more traditional software, often contains two distinct codebases
- One repo for training, one for inference

```
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```




## 11. Deployment

### Algorithmia credentials

If you haven't already, you can signup on Algorithmia here. When creating your API key, make sure to enable `Read & Write` for `Data Access` and enable `Management APIs`.

In [24]:
```
%run -i "deploy.py" -k "simXXXXXXXXXXXXXXXXXXXXXXXXX" -u "b" -a "movie_review_sentiment" -m "model.py" -d "model_requirements.txt"
```

```
Algorithm namepace: b/movie_review_sentiment
Algorithm URL: https://algorithmia.com/algorithms/b/movie_review_sentiment
Pushing source code upstream, uploading model file & compiling algorithm...
Algorithm Source Code is available at: https://algorithmia.com/algorithms/b/movie_review_sentiment/source
Testing new compiled algorithm via API endpoint...
Test complete!
Publishing and deploying algorithm...
Algorithm has been deployed!
```

You just successfully deployed your Tensorflow model!

You can call this API endpoint from anywhere: your mobile app, your website, and more...

# ML deployment has unique challenges...


https://www.flickr.com/photos/3336/27830149309

Data

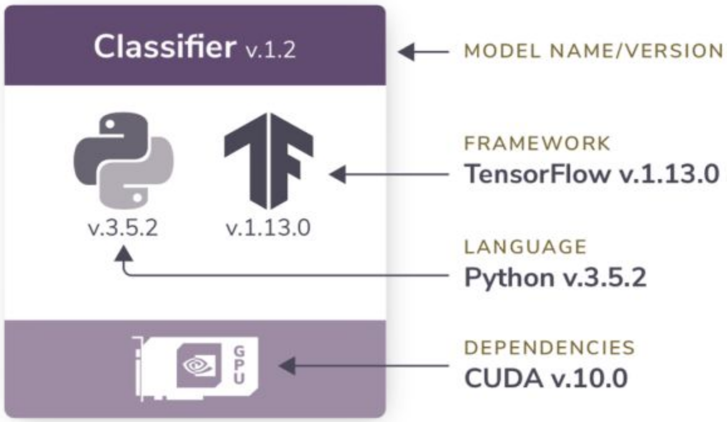
Train

Deploy

Manage

## Docker container

## Docker container



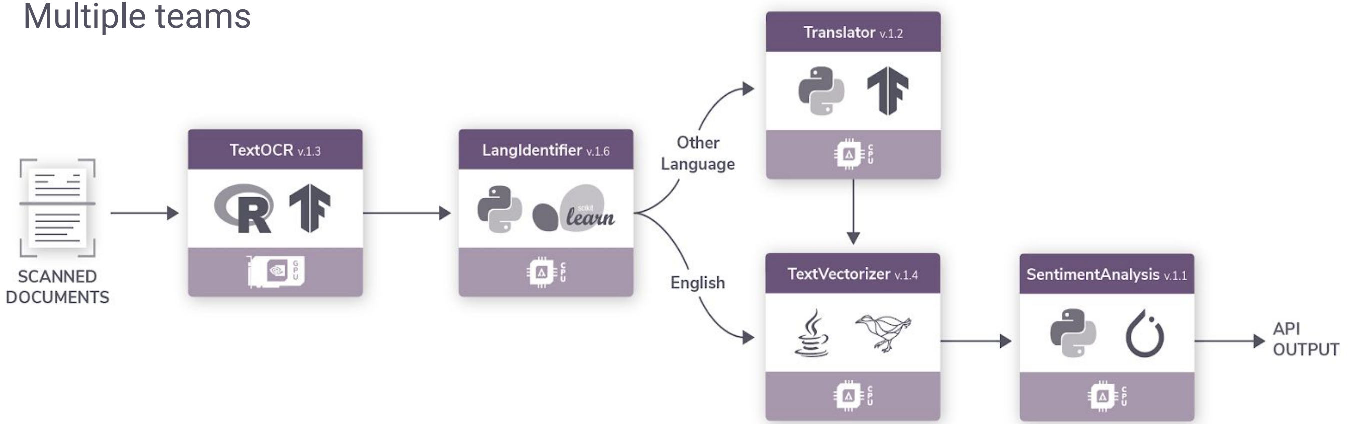

Heterogeneous tooling and dependencies

- Dozens of language/framework combinations
- Hardware dependencies (e.g. CUDA) require substantial architecture investment
- New frameworks emerge every year
- Frameworks and languages evolve constantly, requiring ongoing maintenance and testing

Data · Train · Deploy · Manage

## Composability compounds the challenge

- Multiple frameworks
- Multiple languages
- Multiple teams



Data — Train — Deploy — Manage

| | |
|---|---|
| ACCURACY (TP+TN)/(TP+FP+FN+TN) | How many subjects did the model classify correctly? |
| SENSITIVITY/RECALL TP/(TP+FN) | How many of the actual positives did the model identify correctly? |
| SPECIFICITY TN/(TN+FP) | How many of the actual negatives did the model correct-ly identify? |
| PRECISION TP/(TP+FP) | How many identified as positives were actual positives? |

Measuring model performance

- Success and performance are very context-sensitive
- Multiple success factors
- No one model is right for every job

Where are the tests??

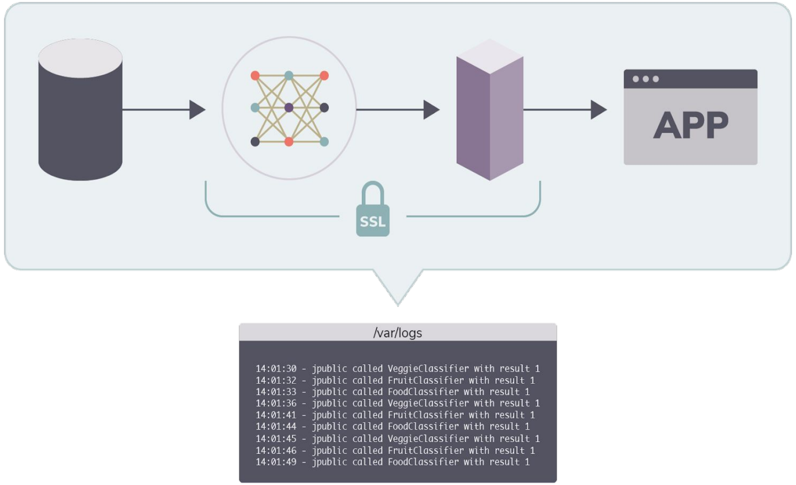
- Unit tests
- Integration tests
- Smoke tests
- Performance tests
- Stress tests

Drift

- Model
- Concept

A/B tests

Data — Train — Deploy — **Manage**




Auditability and governance

- Internal model usage difficult to track across multi-model pipelines
- Auditability and access are major security, compliance concerns


Model drift


Model drift

## Introspection

- Logging
- Monitoring
- Alerting
- Observability

# Deploying ML today is economically challenging

- ✕ Lack of process
- ✕ Wrong incentives
- ✕ Wrong teams
- ✕ Wrong technology
- ✕ Lack of proper champions

# How to tackle

1. Stuck in the lab
2. Disconnected teams
3. Technology mismatch
4. Stakeholder buy-in
5. Hidden technical debt

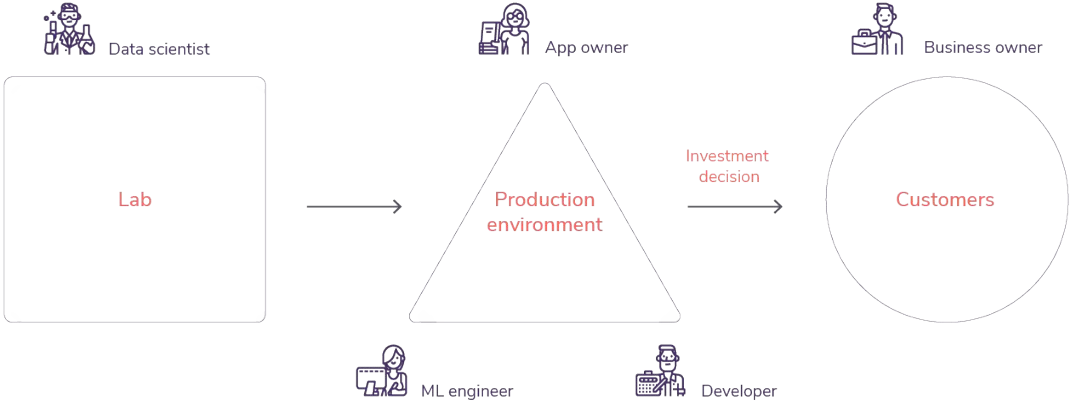
 

# Stuck in the lab

- You decide to invest in data science and AI for competitive advantage
- Proofs of concept funded and experiments running
- Models are trained and concepts are demonstrated but never deployed
- The business is left wondering when results will be delivered

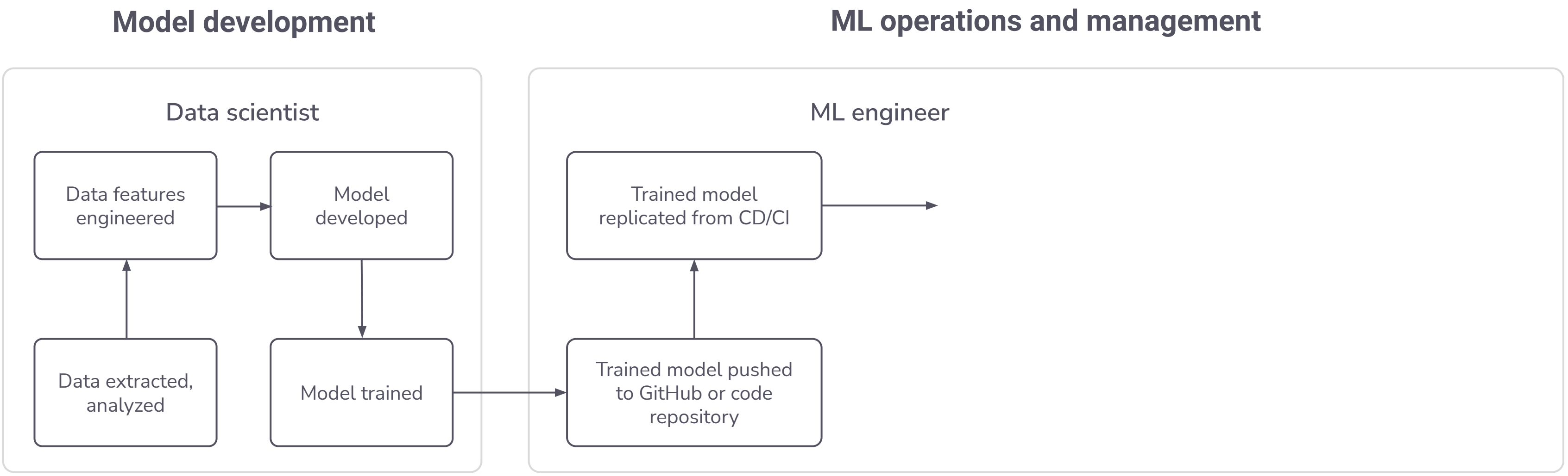
**Must answer:**

- Who funds production?
- Who needs to be involved?
- How does production work in my enterprise?



# Disconnected teams

- Asking data scientists to build infrastructure
- Teams with lack of DevOps experience
- Not partnering the right skill sets inside the org.



# Who is responsible?

- Data scientists
- DevOps
- Business/Executives

**Conway's Law**

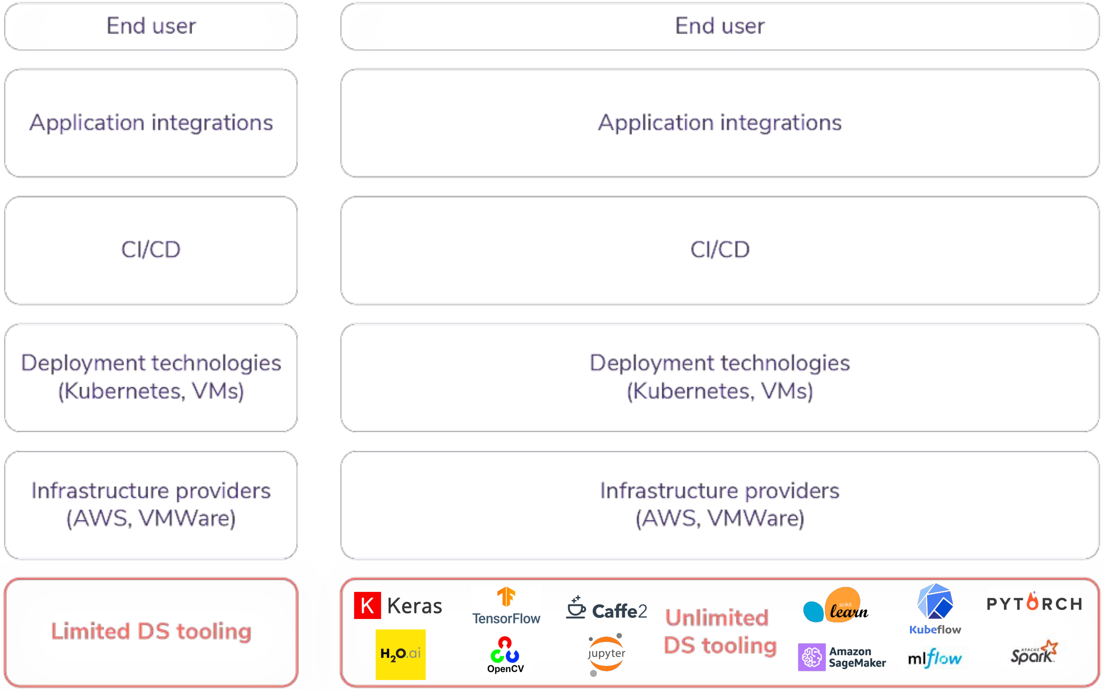
Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.

# Technology mismatch

- Lack of defined technology stack or best practices
- Not building for repeatability, measurability, and auditability
- Proprietary lock-in to tooling
- Not thinking about access to data
- Differences between prod and dev

**Must answer:**

What is the best ML architecture for my organization?



# Stakeholder buy-in

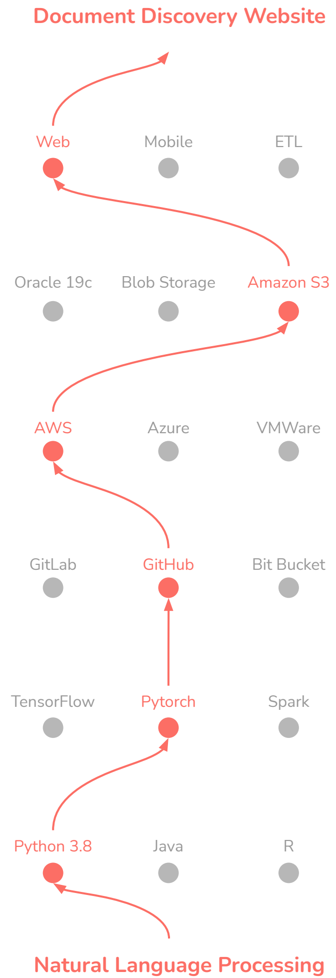
- Like with any new tech deployment, a lack of champions can be detrimental
- ML projects without executive sponsorship rarely see the light of day

> **Must answer:**
>
> How do I get buy-in from stakeholders?

*"Like any introduction of new ideas, tools, or processes, it creates a level of uncertainty due to skepticism, unfamiliarity, or misunderstanding. Fear of failure gets into the way of important and rational decisions."*

—Ian Xiao, Deloitte Manager and AI thought leader

Algorithm to Application



# Lack of process

- Easy to get proofs of concept funded and experiments running
- Once results shown… then what?
- Who funds production, who needs to be involved, how does production work at my enterprise?

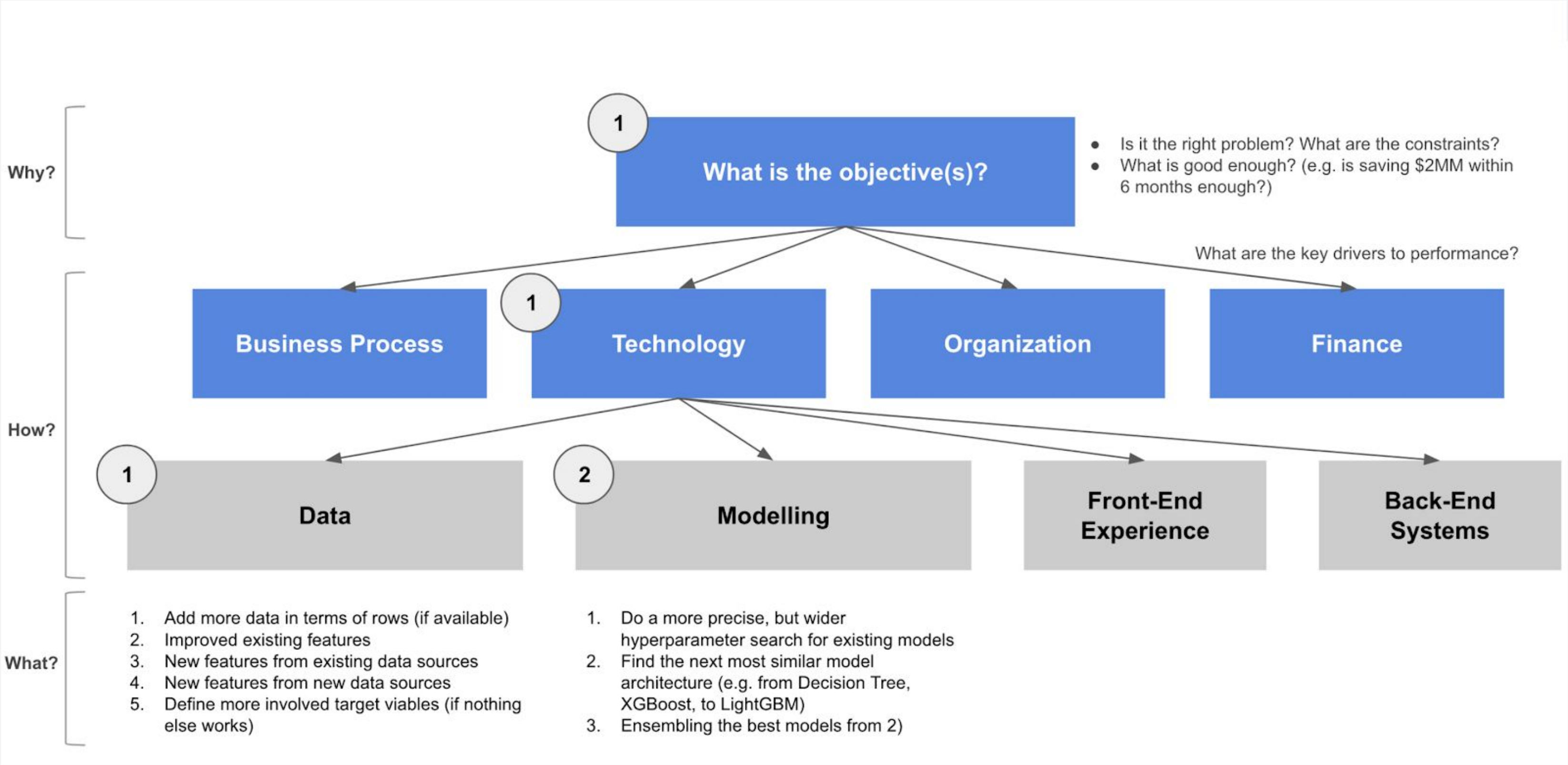
Must be able to answer: How do we go from POC to Production?

**Solution:**

- Plan and fund deployment upfront
- Set clear deployment criterias
- Bring in stakeholders from IT and Devops early
- Build for repeatability in process

# Minimal Justifiable Improvement Tree




Source: ML is Boring - Ian Xiao

# ML is in a huge growth phase, but it is difficult and expensive for DevOps to keep up

- A few models, a couple frameworks, 1-2 languages
- Dedicated hardware or VM Hosting
- Self-managed DevOps or IT team
- High time-to-deploy, manual discoverability
- Few end-users, heterogenous APIs (if any)
- Each algorithm: 1 to 1,000 calls/second, a lot of variance
- Need auto-deploy, discoverability, low (10-15ms) latency
- Common API, composability, fine-grained security

# Hidden technical debt

It is difficult and expensive for DevOps to keep up with the rate of machine learning

- A few models, a couple frameworks, 1-2 languages
- Dedicated hardware or VM hosting
- Self-managed DevOps or IT team
- High time-to-deploy, manual discoverability

- Few end-users, heterogenous APIs (if any)
- Each algorithm: 1 to 1,000 calls/second, a lot of variance
- Need auto-deploy, discoverability, low (10-15ms) latency
- Common API, composability, fine-grained security

> **Must answer:**
>
> What is best way to do MLOps
>
> and management at scale?

# Considerations for ML in the Enterprise

- Infrastructure-agnostic deployment
- Collaboration and pipelining
- Performance SLAs
- Regulatory compliance
- Governance
- Accounting/chargeback tracking
- Security/authentication

# Navigate common pitfalls

- Don’t reinvent the wheel
- Outcomes, not process
- Don’t try to be perfect
- Say no to lock-in
- Tools aren’t solutions
- Audit honestly, revise constantly


**Peter Skomoroch**
@peteskomoroch
Follow

"Big companies should avoid building their own machine learning infrastructure. Almost every tech company I talk to is building their own custom machine learning stack and has a team that’s way too excited about doing this." - @l2k dropping ML knowledge

**Weights & Biases - How to Build a Machine Learning Team ...**
I’ve seen a fair number of the same mistakes over and over again.
wandb.com

3:49 PM - 12 Nov 2018

# Machine Learning != Production Machine Learning

We've spent two decades learning best practices in software engineering, and we can take those lessons and apply them to machine learning.

It's about more than just machine learning, it involves all the components around it, and picking the right tools for the job.

# Thank you

**Further reading and credits:**

- Last defense in another AI winter - Ian Xiao
- Foundations for ML at Scale - Peter Skomoroch
- Hidden Technical Debt in Machine Learning System - Google
- The Roadmap to Machine Learning Maturity - Algorithmia