digital futures lab

Ops & Strategy Planning for AI

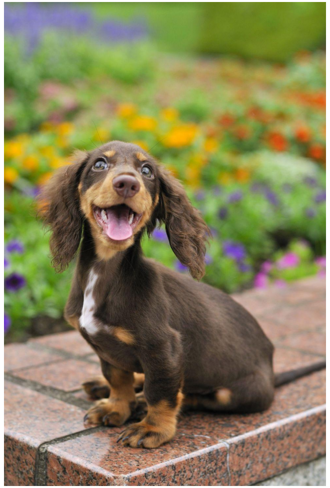
Soma S. Dhavala

Agenda

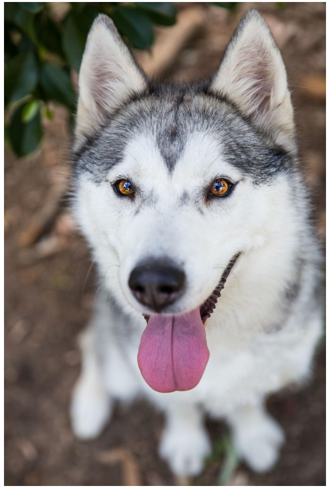
1. Life Cycle of App Development and some useful templates to document them
2. "Questions to Ask" at each stage, with concrete examples of how internal teams responded to these questions, made decisions and dealt with the outcomes
3. Good practices when figuring out the problem that you're trying to address with the AI intervention - questions to ask, things to do, thresholds to obtain/maintain to go forward with the intervention
4. Some guiding principles to find answers or make choices
5. Assessing the impact of a solution using several metrics
6. How to develop feedback mechanisms iteratively and responsive to the end user
7. Putting together the appropriate, interdisciplinary team
8. How to stay agile and flexible around AI resource planning as an organisation

Stakeholder Expectations

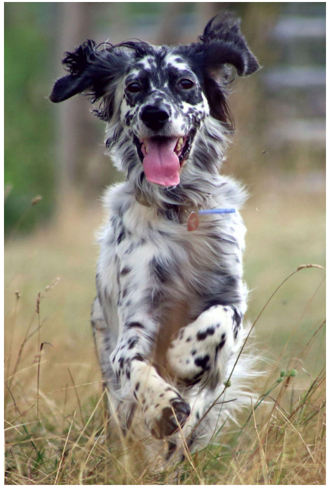
ML team
highest accuracy



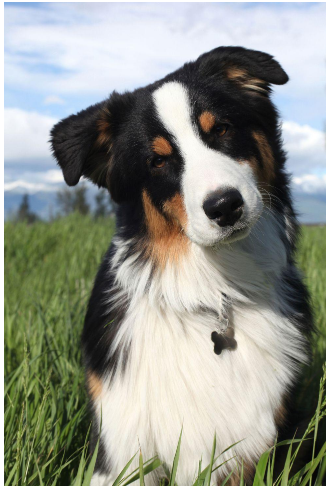
Sales
sells more ads



Product
fastest inference



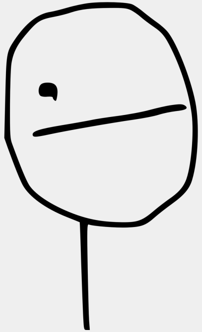
Manager
maximizes profit
= laying off ML teams



[Fig. Source]

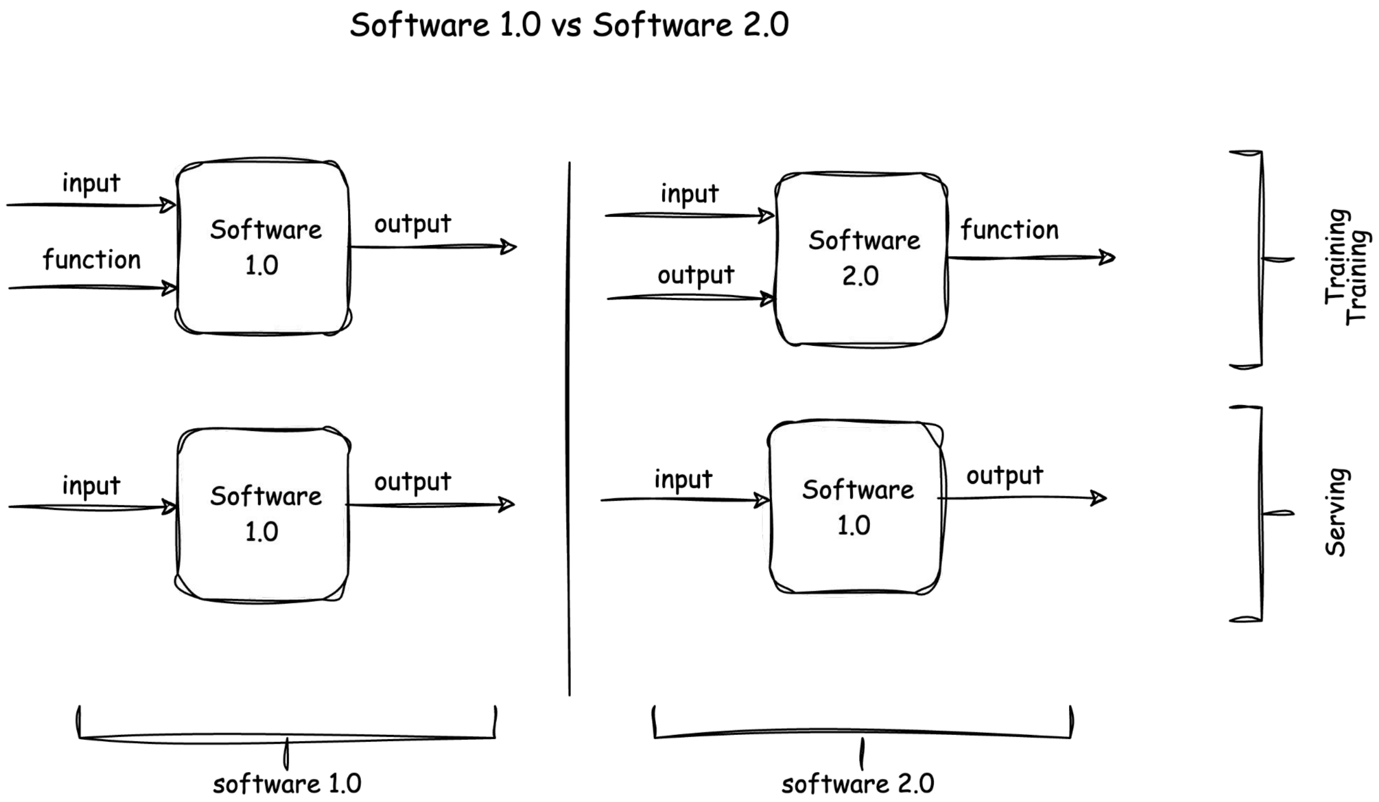
Expectation

Reality



[ML²] SQUARE

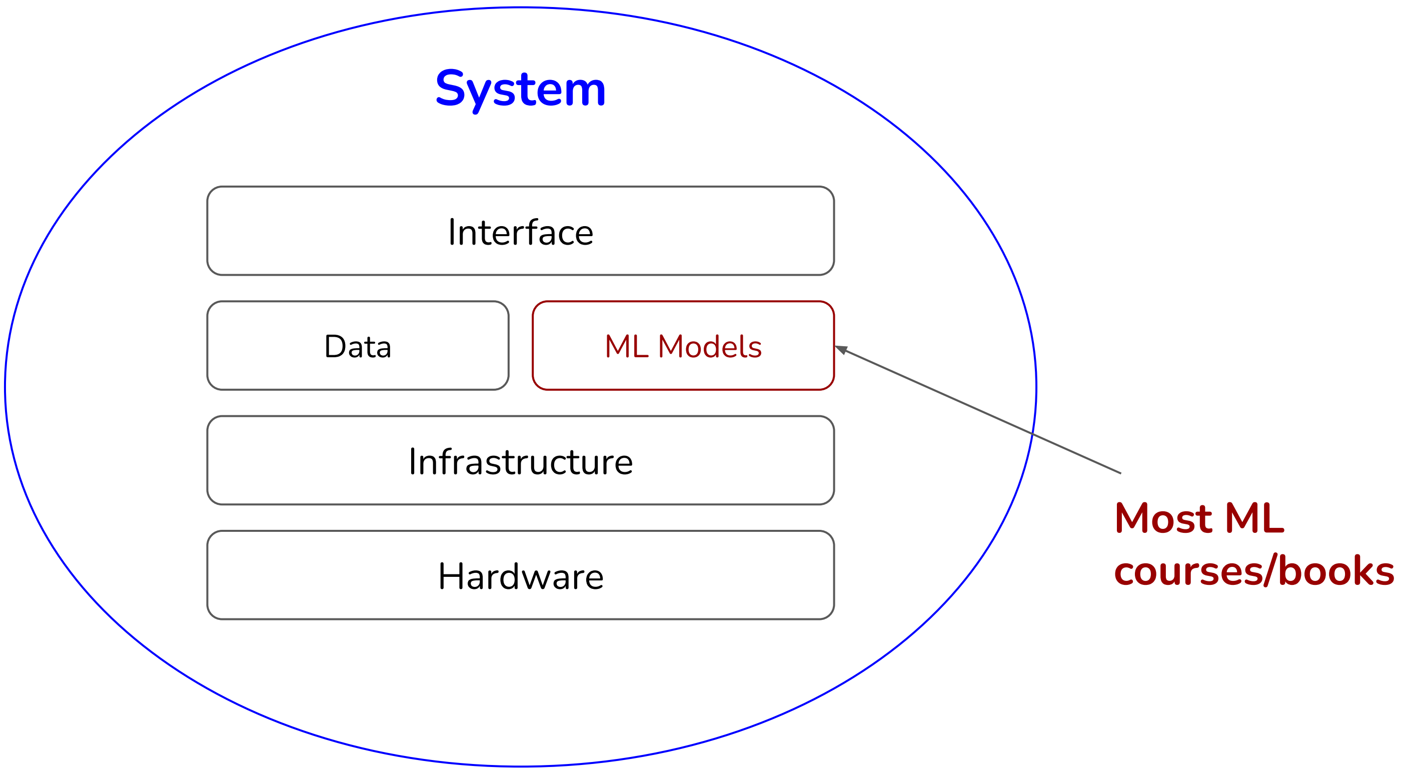
But why?


Software 1.0 vs Software 2.0
input
function
Software 1.0
output
input
Software 1.0
output
input
output
Software 2.0
function
input
Software 1.0
output
Training
Training
Serving
software 1.0
software 2.0

But why?

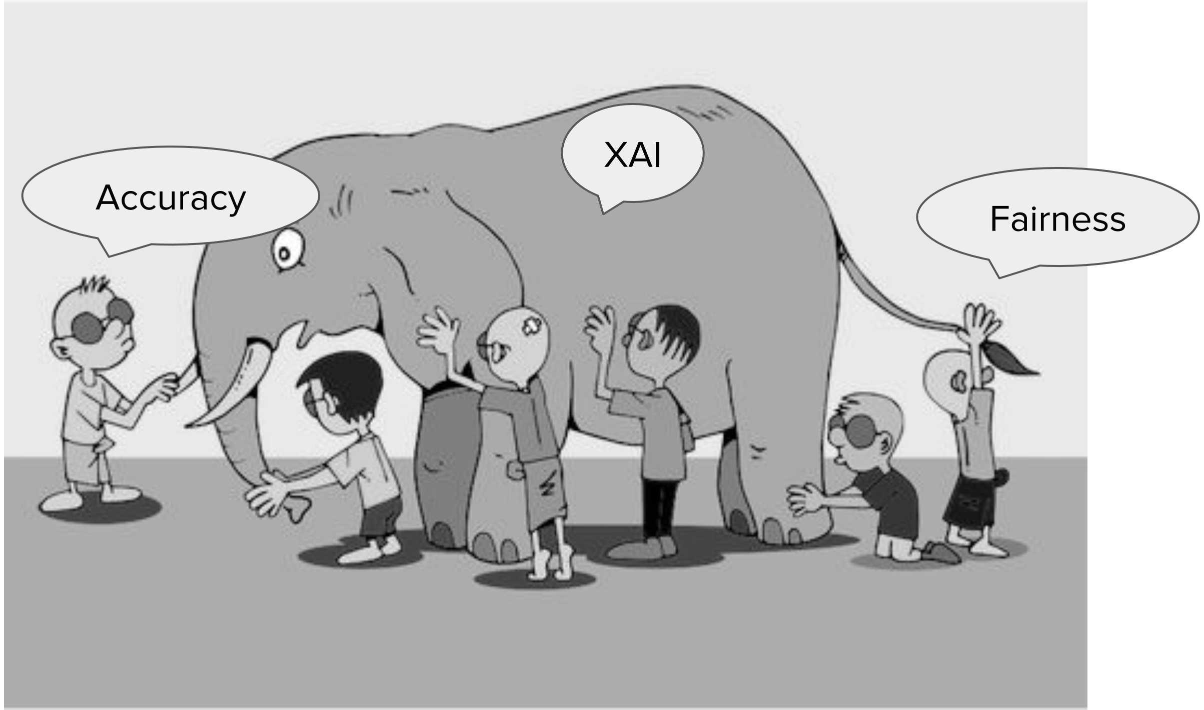
	Software 1.0	Software 2.0
Codified in	Formal Language	Weights & Biases (parameters)
Developed by	Programming	Training
Specification	PRD/SRD	Data
Behaviour	Deterministic	Stochastic
	Provably correct	Provably wrong
	Debuggable	Hard to Debug
	Verifiable	Hard to verify
	Explainable	Hard to explain
	Fixable	Hard to fix
	Idempotent	Hard to reproduce

But why?



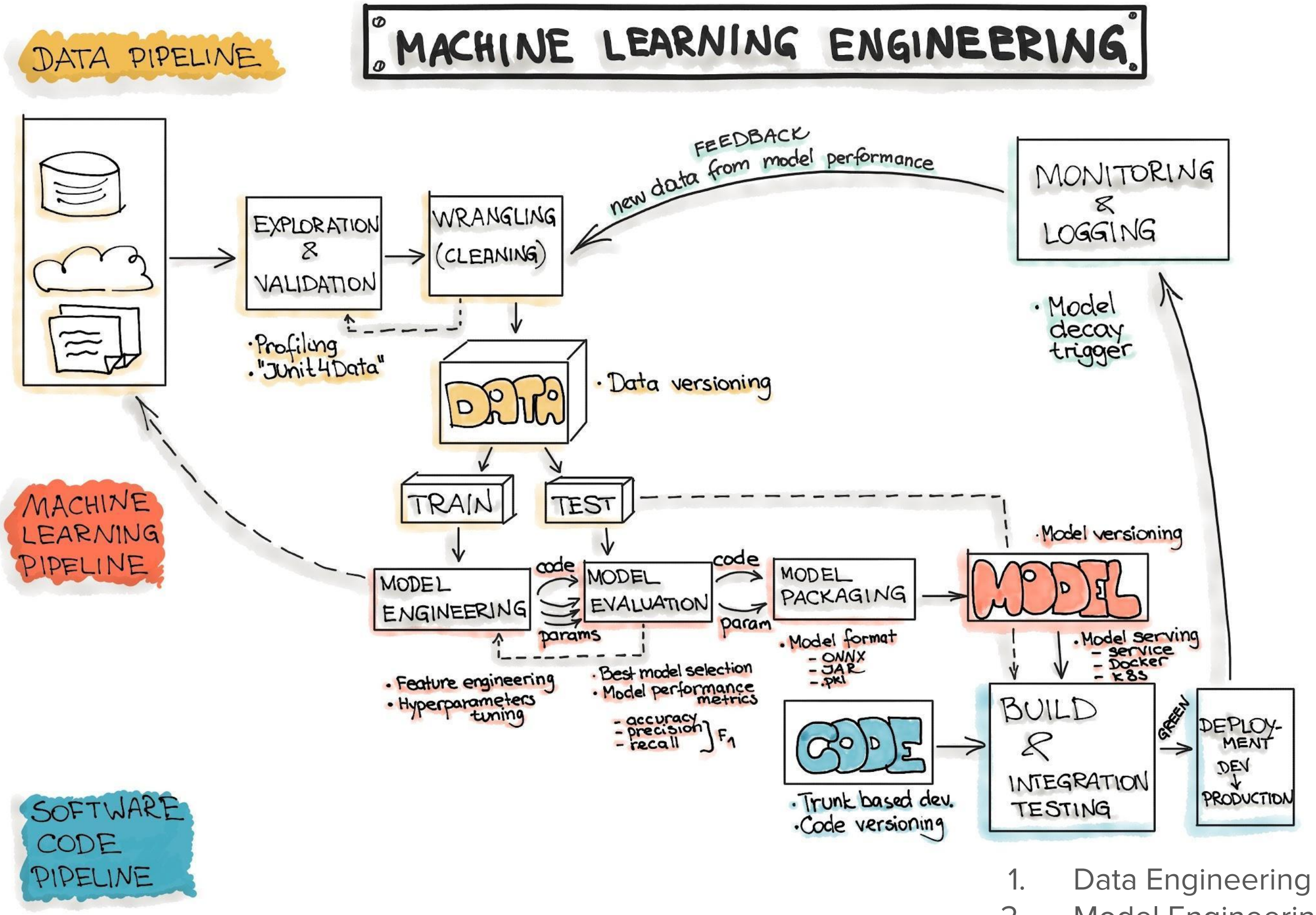
Think systems, not models. Think modeling, not models.
Combating this intellectual inertia is hard.

But why?



Semi-orthogonal sub-fields.

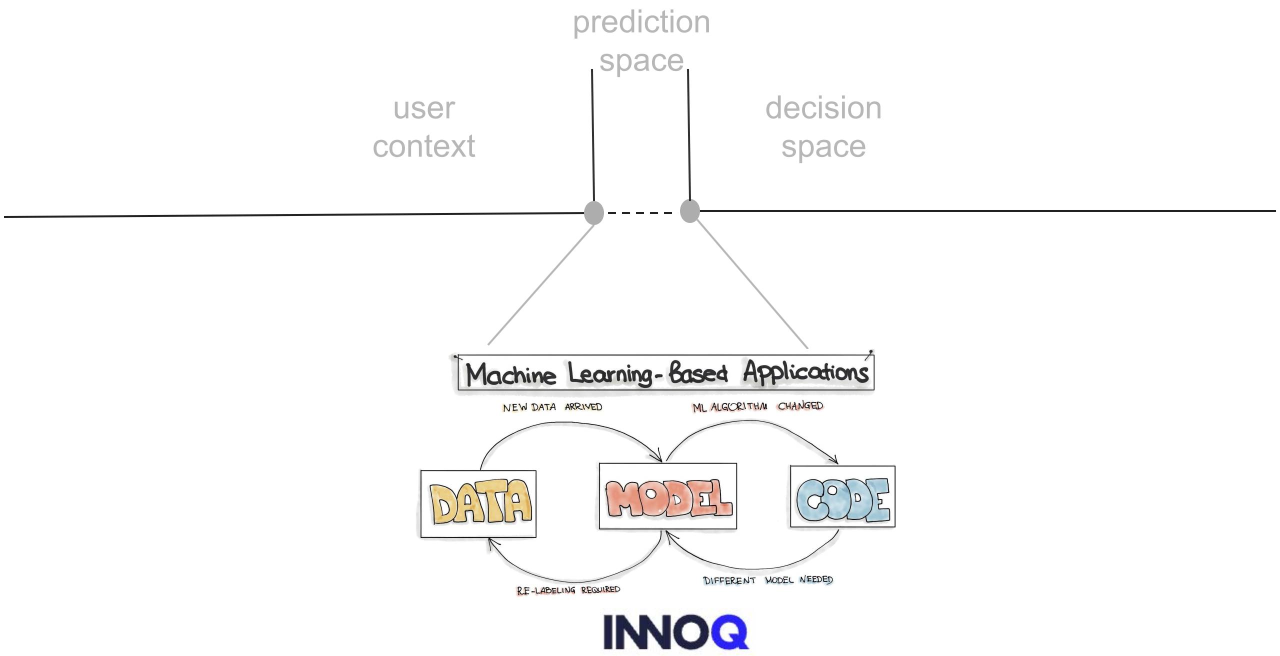
But why?



1. Data Engineering
2. Model Engineering
3. Model Deployment

Hidden Tech Debt

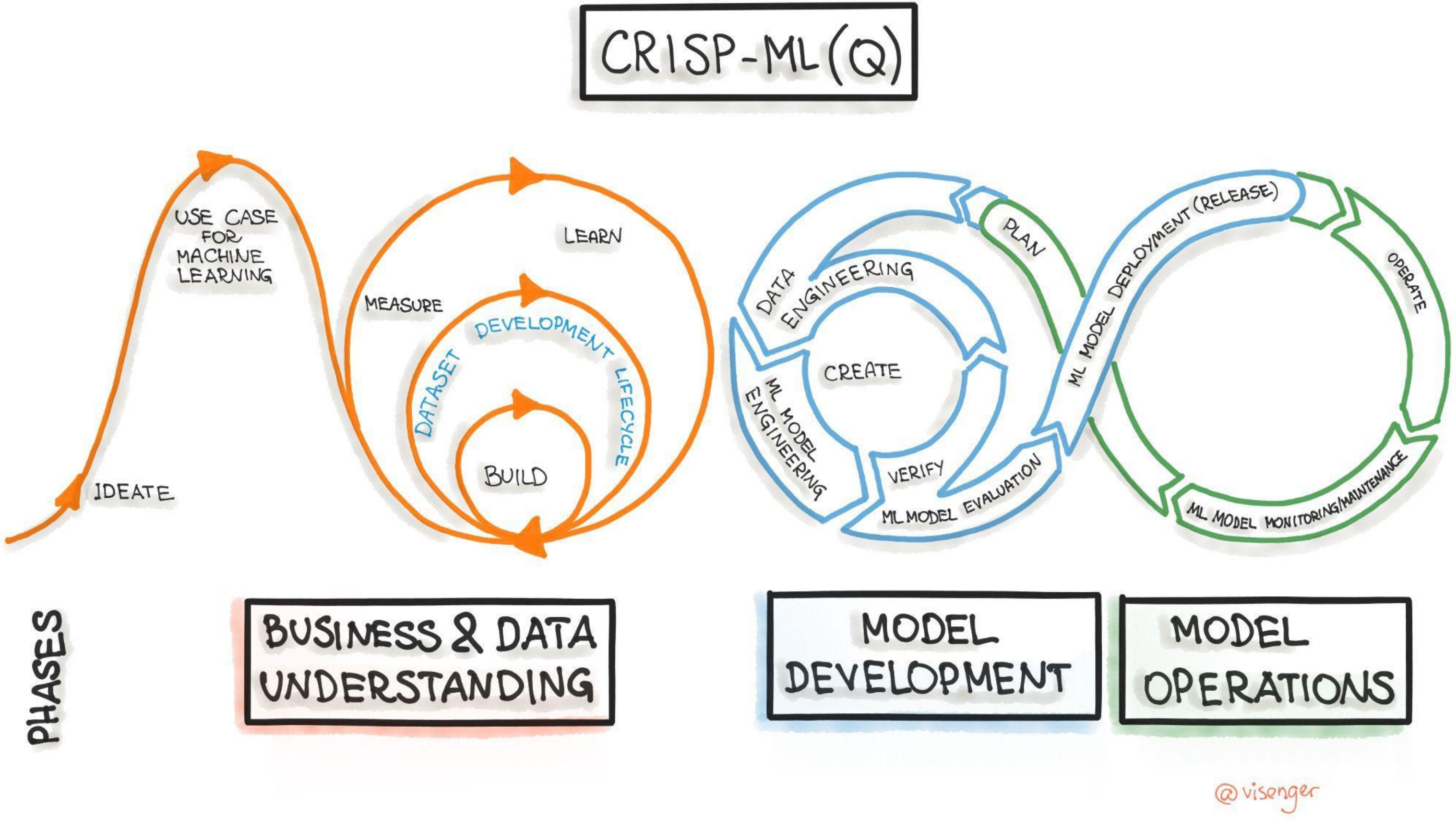
But why?



anything changes > everything changes
premature optimization of a “narrow|deep” problem is the curse to agility

ML Life Cycle

Questions to Ask At Each Stage



CRISP-ML(Q)
USE CASE FOR MACHINE LEARNING
IDEATE
MEASURE
LEARN
DATASET DEVELOPMENT LIFECYCLE
BUILD
DATA ENGINEERING
CREATE
ML MODEL ENGINEERING
VERIFY
ML MODEL EVALUATION
PLAN
ML MODEL DEPLOYMENT (RELEASE)
OPERATE
ML MODEL MONITORING/MAINTENANCE
PHASES
BUSINESS & DATA UNDERSTANDING
MODEL DEVELOPMENT
MODEL OPERATIONS
@visenger

Fig. Source

Stage	Project Lead
Discover the problem	Identify “who”it is for?, “what” it is the problem (not the solution)? What are the business metrics? Can it be solved? Is AI necessary? Data is from the past. Is that the future we want to extrapolate? What is/will be the RoI? (put in real monetary terms of A/B). What all data can be collected/ needs to be collected?
Build an MVP	What is the scope which is simple (and testable) but not simpler. Can it still be functional?
Pilot it	What is the efficacy? Is it working? Verify assumptions from MVP. A good DoE helps.
Launch it	Can we lunch, scale? What needs to improve? Can it sustain? How to maintain it?

Broadly four stages. Iterate.

ML Project Card: A QA format document

Purpose:
1. Help the solution & product team think thoroughly about the problem
2. Serves as communication aid
3. With versions, can see the progress and evolution

Then:
1. Design Thinking to understand the problem
2. Discover the problem, run through, develop a solution, test, iterate
3. Express it using Project Cards

Business Structure
1. What is the problem?
2. Who it is for?
3. Why it needs to be solved?
4. How will the solution look like (workflow and a mental model)?
5. What will it lead to?
6. What are the risks?

ML Solution and Execution
1. What is the **prediction** problem?
2. How the objective will be measured?
3. How will it be tested?
4. Data: What kind, how and how much and what for?
5. What is the roadmap/plan?
6. What resources are needed?

building on Data cards, Model cards, MLOps cards

source

Best Practices

and [some] guiding principles

Principles/ Heuristics:

1. Imperfect solution to a right problem > perfect solution to a wrong problem
    a. Breadth-first. Not depth-first
2. Premature optimization is the curse of ML (and any field)
    a. Start simple.
    b. Simple, low-tech debt solution to complex but highly performant
3. No perfect launch. It is a journey.
    a. Iterate quicker. Fail faster.
    b. Prioritise where to improve, not what you think is important
4. “So what” > Why > What > How (solution)
    a. Insist on “So What?” seven times.
    b. But we often start with a solution (or a technology)
5. Err on “data” side.
    a. Collect more but purpose driven (even if the purpose is anticipatory).
6. Outputs will be wrong
    a. Abstain when not sure (can the UX support it?)
        i. Models need not make decision all the time
        ii. Abstention is a lever, in addition to “scope (functionality)”, “resources (time, compute, human)”, “quality”.
        iii. Design is about tuning these four degrees of freedom via negotiation.
7. Not all decisions are equal.
    a. Cost-aware decision making.

Handle ambiguity NOT just business ambiguity, BUT user-experience ambiguity as well.

Metrics

Metrics Categorization

- ML Metrics > Business Metrics
- Specific > Universal
  - **RMSE of a CYE model** > Reduction in Risk by 10% > Profit by 5% > Improved Quality of Life **> Happiness**

North Stars - Services

- Affordability: Is the service/product affordable
- Availability: Is the service/product available for “all” (scale) and “all the time” (reliability)
- Accessibility: Is the service/product accessible

dictated by **who** it is for

- Coverage: Does the service/product has high coverage?
  - Eg: a PHC offers a Screening solution but not Diagnostics!
  - # of auxiliary services in the suite, as a proxy

dictated by **what** is being built?

- Leverage
  - How many new opportunities it can unlock
  - The more foundational, context-free the service is, the more leverage it can generate

dictated by **where** in action cascade it is?

North Stars - Solutions

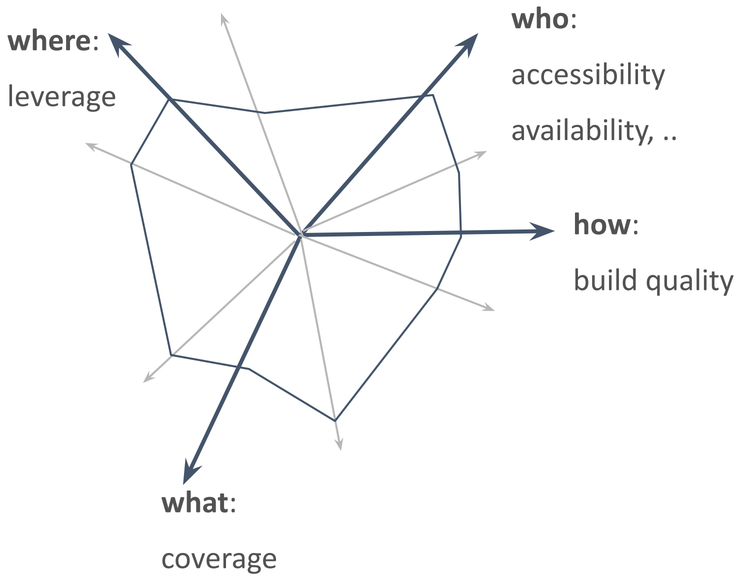
- Performance
- Inclusivity, Observability, Auditability
- Trustworthiness, Fairness
- Modularity, Extendability, Interoperability

dictated by **how** a solution is built?

Instrument and enable them
It is a journey from: ML to Impact, Specific to Universal, Direct (Observable) to Indirect, Individual to Societal.

From Prediction Machines to Decision Engines

- ML is a glorified auto fill technique
- ML is a “means” to an “end’, not an “end” in itself
- ML is like any other piece of technology
- No distinction is made among {ML, AI, Statistics, Data Science}
- Predictions alone are not sufficient
- They must enable decision making
- Important to ask: “so what”
- Follow the trail of actions



One’s information is somebody else’s Intelligence
This is, perhaps, the only way to create/unlock value
Any project management decision must lead to unlocking or creating that value.

An overall score can be given to each solution, with different weightages to each dimension

Feedback

Unknown unknowns

Describe the "context" as elaborately as possible

- Use "Dimensional Analysis" techniques from the Database world (knowledge engineering)

Log

- Instrument the telemetry and log

Analyze

- Usage (performance) by various dimensions

Improve

- [Business] Redefine the problem (label engineering, but we mostly focus on Models)
- [Marketing] Incentivise usage. Solve auxiliary problems which help the primary problem.
- [ML] Collect different type of data. Retrain.
- [Engg.] Retool. Add additional value-add functionality

Build for observibility (telemetry)
Invest in MLOps (not very hard)

Team

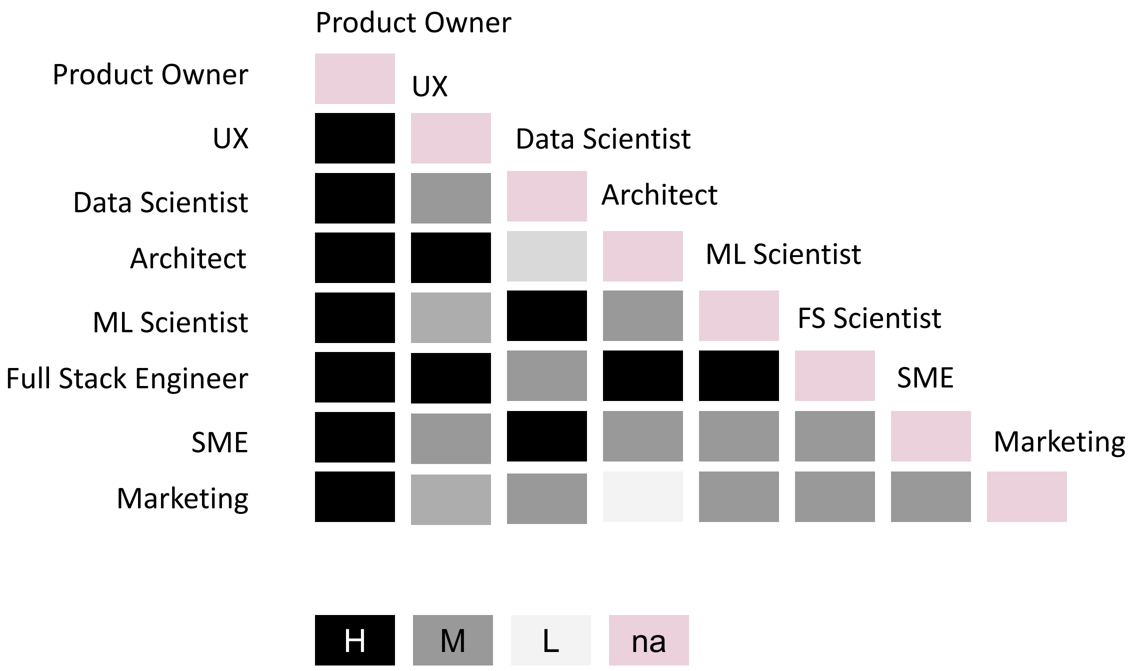
Team: VERY multi-disciplinary

	Discover	MVP	Pilot	Launch	Cares & Concerns
Product Owner	Run	Run	Run	Run	Functional Reqs, Orchestration
UX	Run	Run	Run	Consult	HCD, Workflow, UQ Comm.
Data Scientist	Run	Consult	Consult	Advise	Label & Feature Eng., SIgnal Analysis, Baselines
Architect	Consult	Run	Advise	Consult	Non-functional Reqs (eg Agility)
ML Scientist	Advise	Run	Consult	Consult	Model dev, Metrics, Data Req.
Full Stack Engineer	Advise	Run	Consult	Consult	Utility, Cost, SLA, Ext., Mod. etc
SME	Consult	Consult	Advise	Advise	Problem Selection, Alignment, UX, Comm., Validation
Marketing	Advise	Advise	Consult	Run	Change Making, Adoption Strategies, Awareness

Run | Consult | Advise

Product Owner role is KEY to success

Team: Conway's Law : Team structure surrogate to solution architecture



The org's communication structure should reflect the data/information flow of the solution.

Team:

People (Team) > Problem > Process

Any other ordering is suboptimal

Agility

Principle of Variability

Identify sources of variation and rank them

most variable to least variable (known variability) > Abstract > Customizability

most uncertain to least uncertain (unknown variability) > Building Blocks & Factories > Extensibility

Build for modularity (separation of concerns) and extensibility (new functionality)

Narrow with peripheral vision (for ML)

ML Problem should be as narrow as possible for it to work (narrow intelligence)

But it is at odds with agility

So, clear ML problem definition is important. But collecting “additional” metadata helps you pivot.

Often, this is difficult to iterate upon

Typically, it is UX (onboarding, value-add). Can be offloaded to implementation partner.

Deep empathy > Anticipation > Proactive Measures
How you build mostly dictates agility to react and respond

Expectation

Reality

Let us bridge the gap between Expectation and Reality by

- Understanding how AI differs from traditional tech
- Following a breadth-first approach
- Exercising additional degrees of design freedom!
- Developing empathy for every stakeholder in the solution
- Assembling a multidisciplinary team
- Recognising that People [team] > Problem > Process

Make ML great again :)