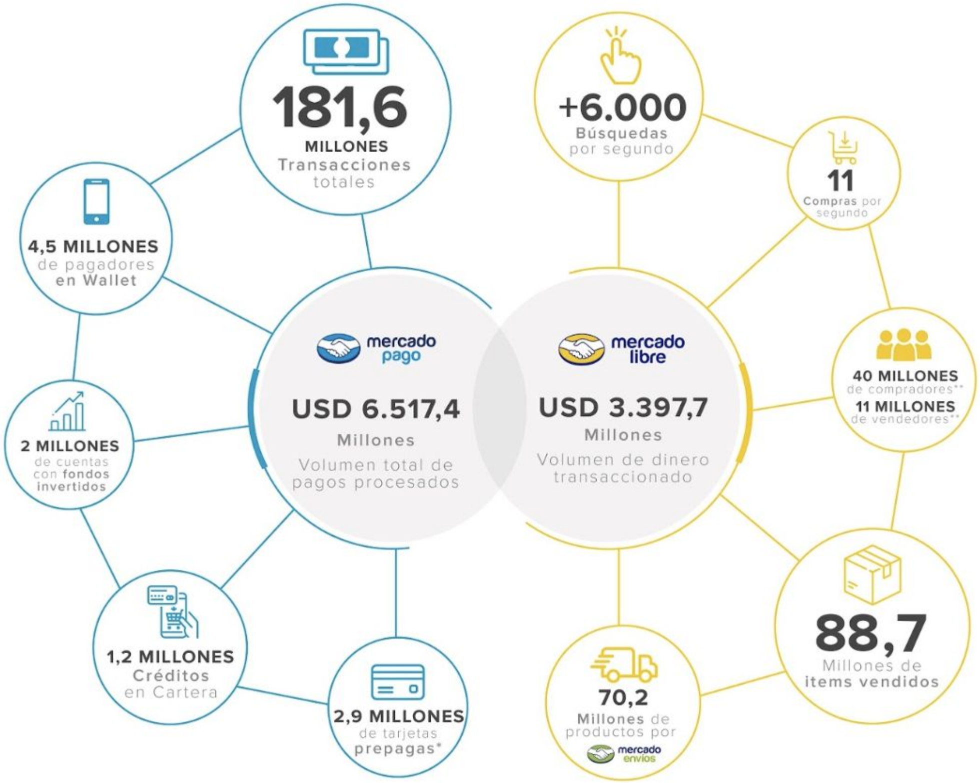
Reliable ML systems

SciPy Latam 2019, Bogotá

[Javier Mansilla - javier.mansilla@mercadolibre.com](mailto:javier.mansilla@mercadolibre.com)
Bogotá, 2019

One ecosystem

Large-scale operations



Introduction and assumptions

what is this talk about?

Introduction and assumptions

what is this talk about?

Imagine a system serving 500k predictions per minute.

8 milliseconds per request.

Re-trained reguraly.

Introduction and assumptions

what is this talk about?

Imagine a system serving 500k predictions per minute.

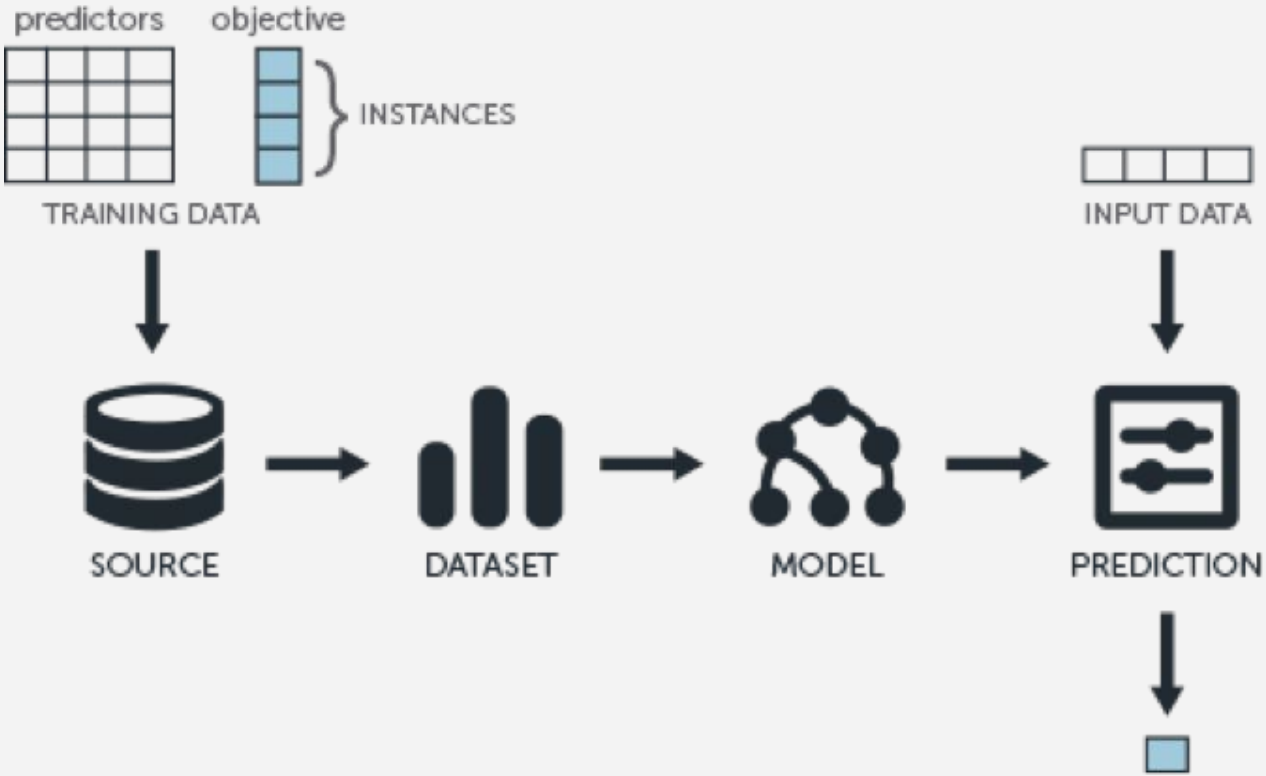
8 milliseconds per request.

Re-trained reguraly.

Pro tip: be kind with the one that's on call at night.

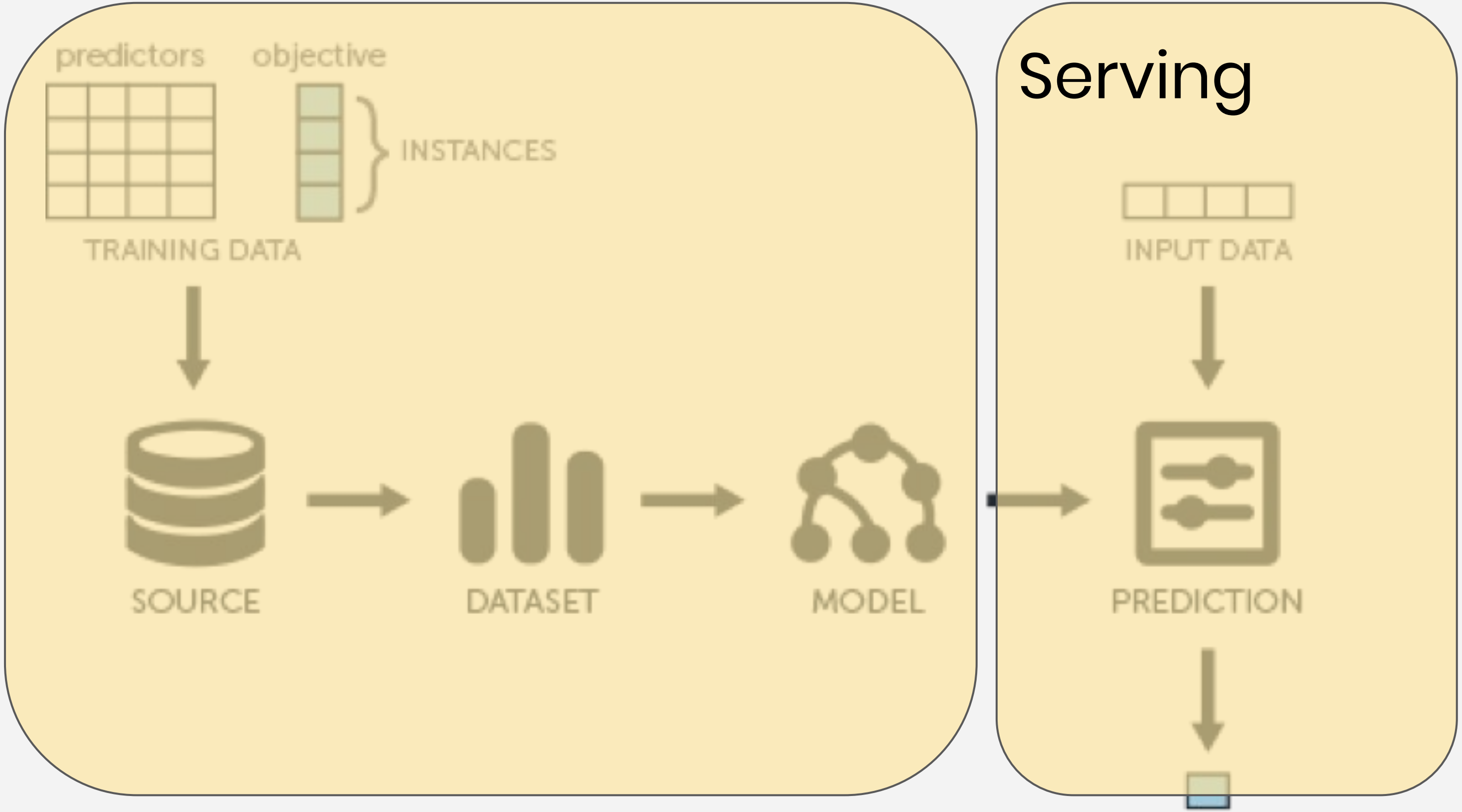
AN APPROPRIATE SIMPLIFICATION

A machine-learning based system



AN APPROPRIATE DIVISION

A machine-learning based system



WHAT DO I MEAN

When I say “testing”...

- Infrastructure
- Data
- Logging
- Metrics
- Monitoring

Code is code

Software engineering best-practices apply

Open your mind to testing

In sync with the industry

Let’s not reinvent the wheel

Features & Data

What's your ML Test Score? A rubric for ML production systems

Eric Breck, Shanqing Cai, Eric Nielsen, Michael Salib, D. Sculley

Proposes practices divided in 4 areas:

- *Features & Data*
- *Model Development*
- *Infrastructure*
- *Monitoring*

PRACTICES

Features & Data

1. Feature Distribution Expectations
2. Feature relationship with target or pairwise
3. Worth paying cost per feature
4. Easy & Stable Feature deprecation
5. Privacy across Pipeline
6. Calendar time for create & add new feature to prod
7. Usual testing practices for features code

PRACTICES

Features & Data

1. Feature Distribution Expectations
2. Feature relationship with target or pairwise
3. Worth paying cost per feature
4. Easy & Stable Feature deprecation
5. Privacy across Pipeline
6. Calendar time for create & add new feature to prod
7. Usual testing practices for features code

PRACTICES

Features & Data

1. Feature Distribution Expectations
2. Feature relationship with target or pairwise
3. Worth paying cost per feature
4. Easy & Stable Feature deprecation
5. Privacy across Pipeline
6. Calendar time for create & add new feature to prod
7. Usual testing practices for features code

PRACTICES

Model Development

1. Code review model specifications
2. Relationship between proxy & actual metrics
3. Explore hyperparameters
4. Effect of model staleness
5. Compare with simpler baseline
6. Quality on relevant data slices
7. Presence of implicit bias

PRACTICES

Model Development

1. Code review model specifications
2. Relationship between proxy & actual metrics
3. Explore hyperparameters
4. Effect of model staleness
5. Compare with simpler baseline
6. Quality on relevant data slices
7. Presence of implicit bias

PRACTICES

Model Development

1. Code review model specifications
2. Relationship between proxy & actual metrics
3. Explore hyperparameters
4. Effect of model staleness
5. Compare with simpler baseline
6. Quality on relevant data slices
7. Presence of implicit bias

PRACTICES

Infrastructure

1. Reproducibility of training
2. Integration test model specification code
3. Integration test full pipeline
4. QA before serving
5. Incremental training
6. Server vs model sync (Canary deploy)
7. Roll back to previous version

PRACTICES

Infrastructure

1. Reproducibility of training
2. Integration test model specification code
3. Integration test full pipeline
4. QA before serving
5. Incremental training
6. Server vs model sync (Canary deploy)
7. Roll back to previous version

PRACTICES

Infrastructure

1. Reproducibility of training
2. Integration test model specification code
3. Integration test full pipeline
4. QA before serving
5. Incremental training
6. Server vs model sync (Canary deploy)
7. Roll back to previous version

PRACTICES

Monitoring

1. Upstream instability in features (in train & serving)
2. Data invariants hold (train & serving)
3. Features compute the same in train & serving
4. Model staleness status
5. Invalid data in t & s (NaN, Infinite, etc)
6. Bare metal peaks or leaks
7. Quality regressions on served predictions

PRACTICES

Monitoring

1. Upstream instability in features (in train & serving)
2. Data invariants hold (train & serving)
3. Features compute the same in train & serving
4. Model staleness status
5. Invalid data in t & s (NaN, Infinite, etc)
6. Bare metal peaks or leaks
7. Quality regressions on served predictions

PRACTICES

Monitoring

1. Upstream instability in features (in train & serving)
2. Data invariants hold (train & serving)
3. Features compute the same in train & serving
4. Model staleness status
5. Invalid data in t & s (NaN, Infinite, etc)
6. Bare metal peaks or leaks
7. Quality regressions on served predictions

Implementation

What do we do with our teams?

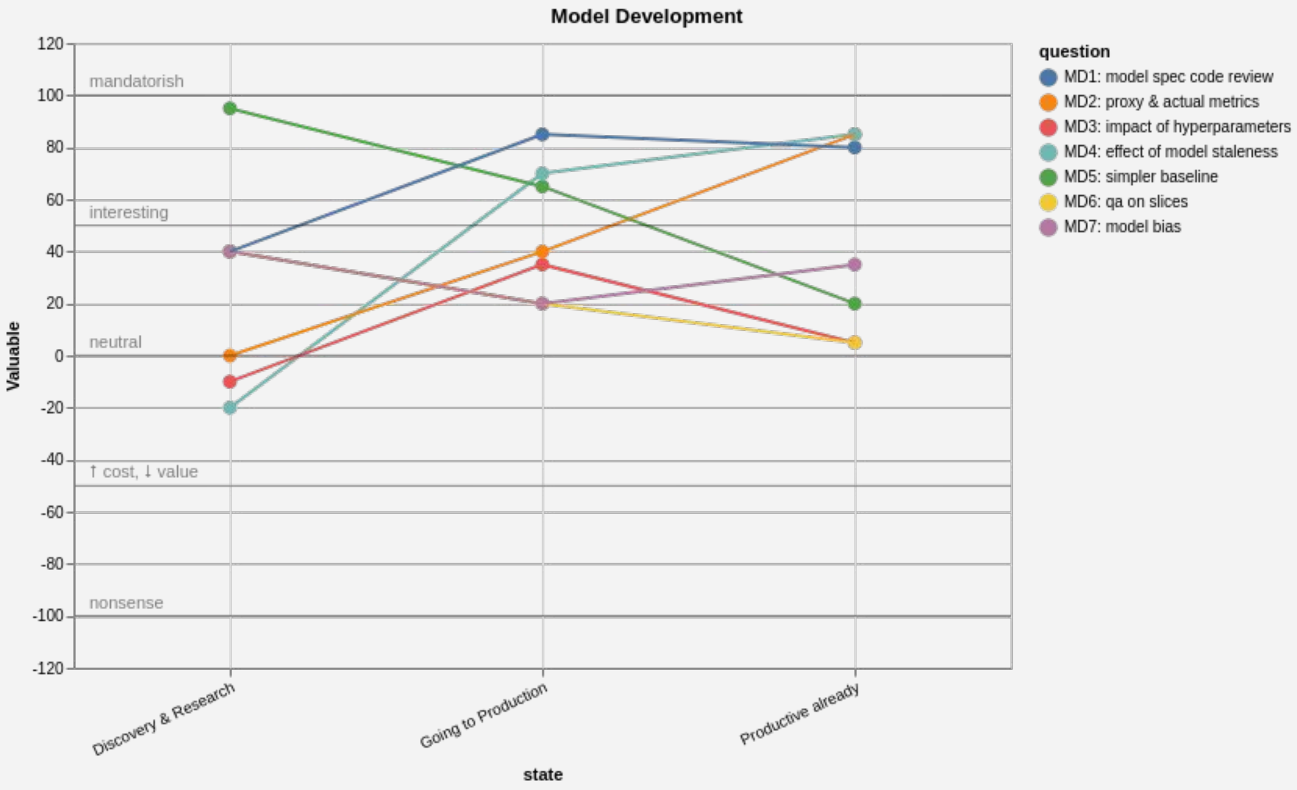
OUR DEFINITION

Project’s stages

1. **Research Only projects / Discovery:**
   the team is working towards validating feasibility, or discovering opportunities
2. **Production in the Roadmap**
   the team is working focused on reaching production in the short-mid range
3. **Already in production**
   the team is maintaining, upgrading and monitoring an initiative that's already in production

OUR DEFINITION

What practices make sense in each stage?



Thank you!

Questions?