Monitoring AI with AI

Credits

hydrosphere.io

Calls everyone CEO but not himself

Like `Matrix`s architect but in ML world

Hacker of company growth

Created DevOps

Knows physics better than my professor

Knows everything better than you, believe me

His thoughts are written in scala

In a couple of years will be able to deploy your mind

Will find you outlier everywhere

Traditional apps vs ML

Unit testing	Model evaluation
(Micro)service	Model as a service
Docker per service	Docker per model
Eng + QA owning a service	1 ML Eng owning 10-20 models
Fail loudly	**Fail silently**
Can work forever if verified	Performance declines / need retraining
App metrics monitoring	Data / model metrics monitoring

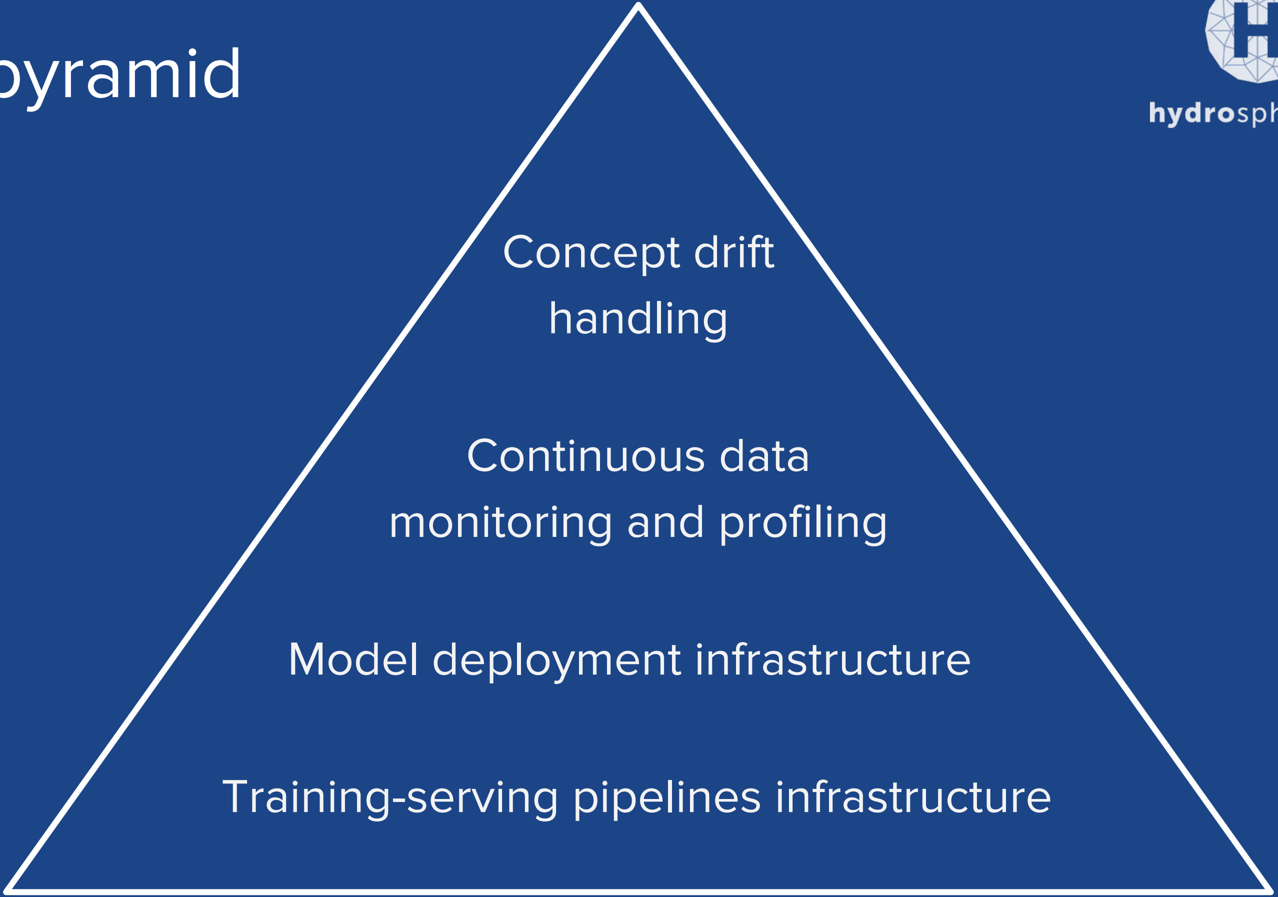
Where may AI fail in prod?

Everywhere!

Why may AI fail in prod?

- Bad training data
- Bad serving data
- Training/serving data skew
- Misconfiguration
- Deployment issue
- Retraining issue
- Performance
- Concept drift
- …

AI reliability pyramid



Concept drift handling

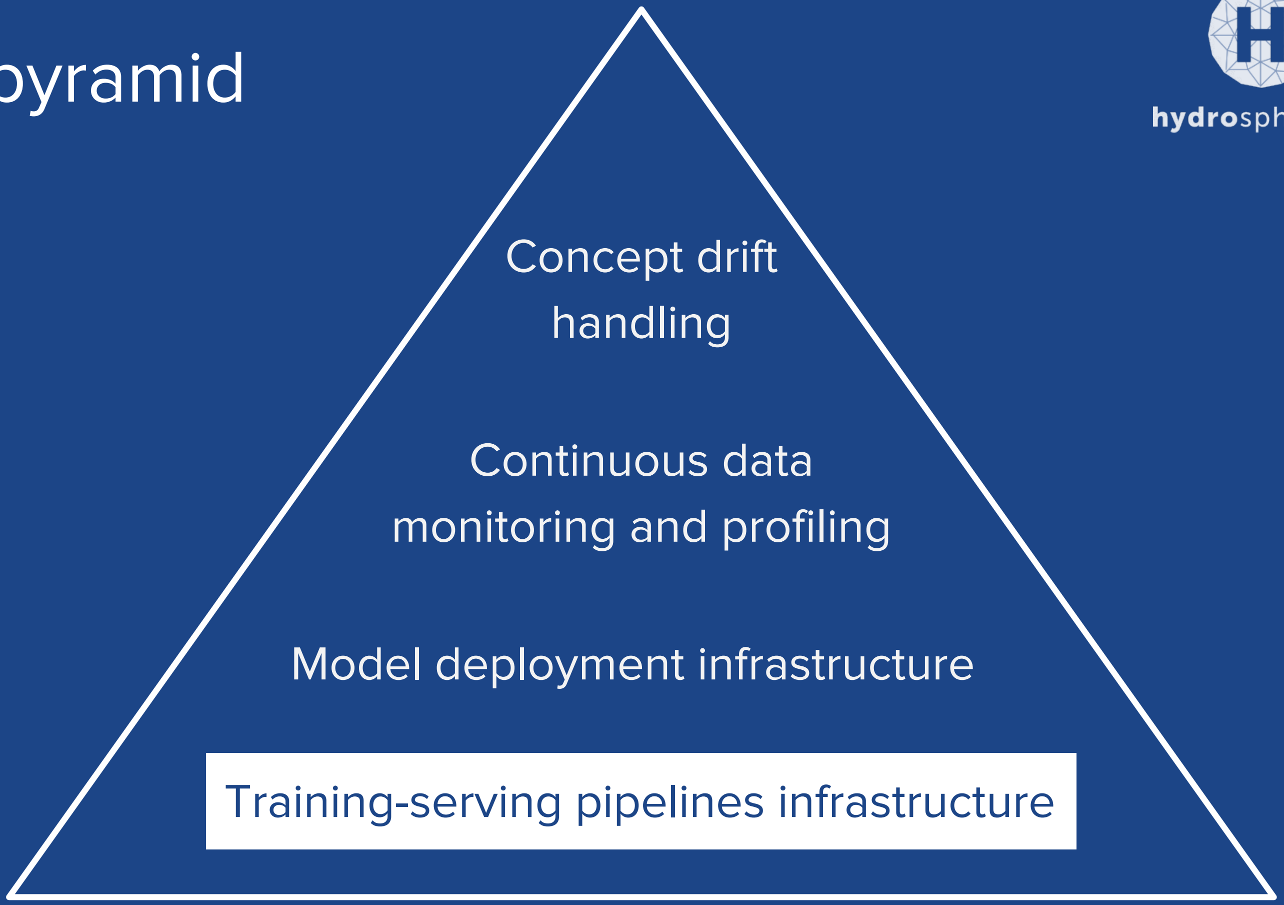
Continuous data monitoring and profiling

Model deployment infrastructure

Training-serving pipelines infrastructure

hydrosphere.io

AI reliability pyramid



Concept drift handling

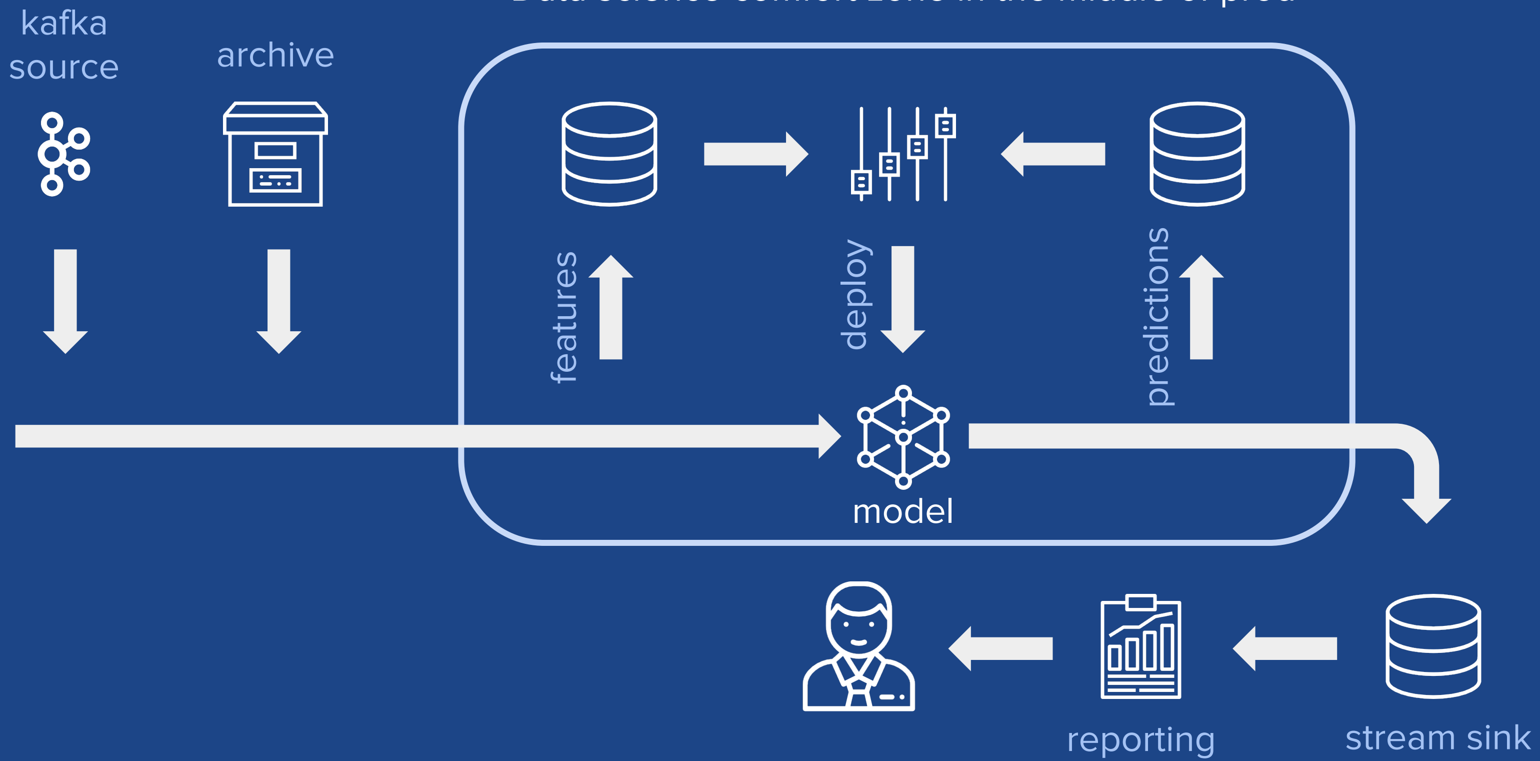
Continuous data monitoring and profiling

Model deployment infrastructure

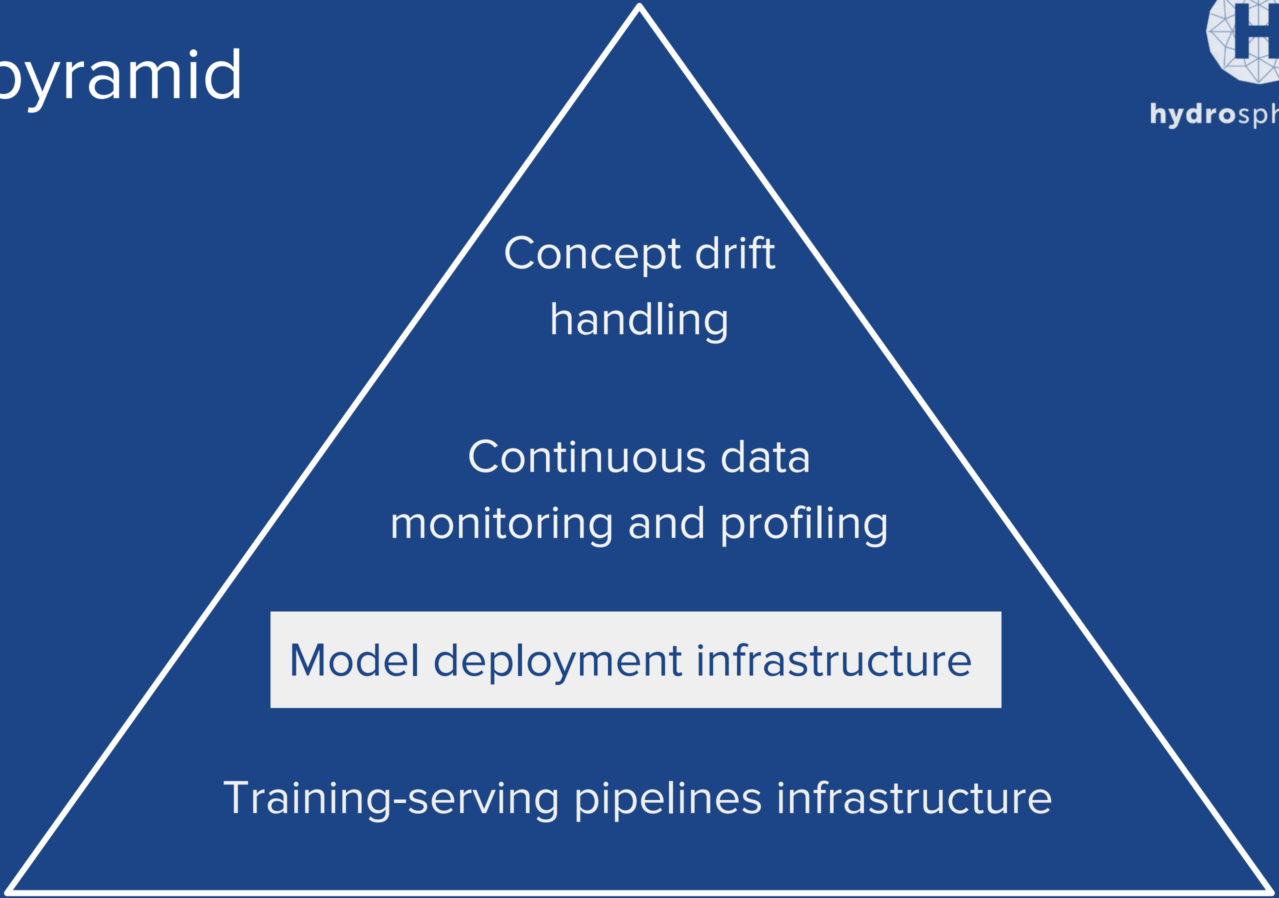
Training-serving pipelines infrastructure

Reliable training-serving pipelines

Data science comfort zone in the middle of prod



AI reliability pyramid



Concept drift handling

Continuous data monitoring and profiling

Model deployment infrastructure

Training-serving pipelines infrastructure

Model deployment and integration

How to integrate it to AI/ML application?

model.pkl
model.zip

Model server

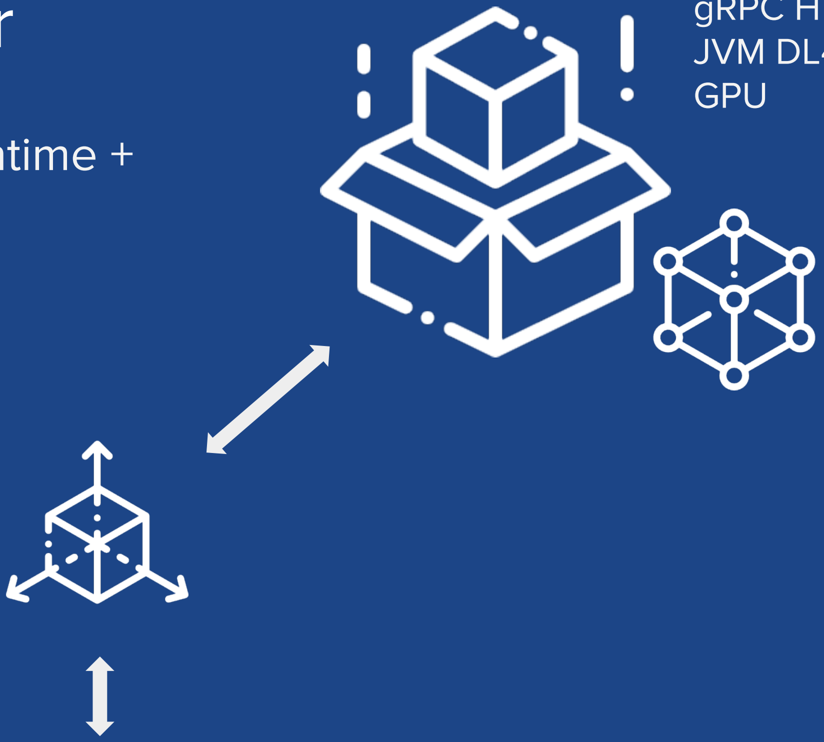
Model artifact + Runtime + Deps

gRPC HTTP server
JVM DL4j
GPU

model v2
[...]

Model server

Model artifact + Runtime + Deps + Sidecar



gRPC HTTP server
JVM DL4j
GPU

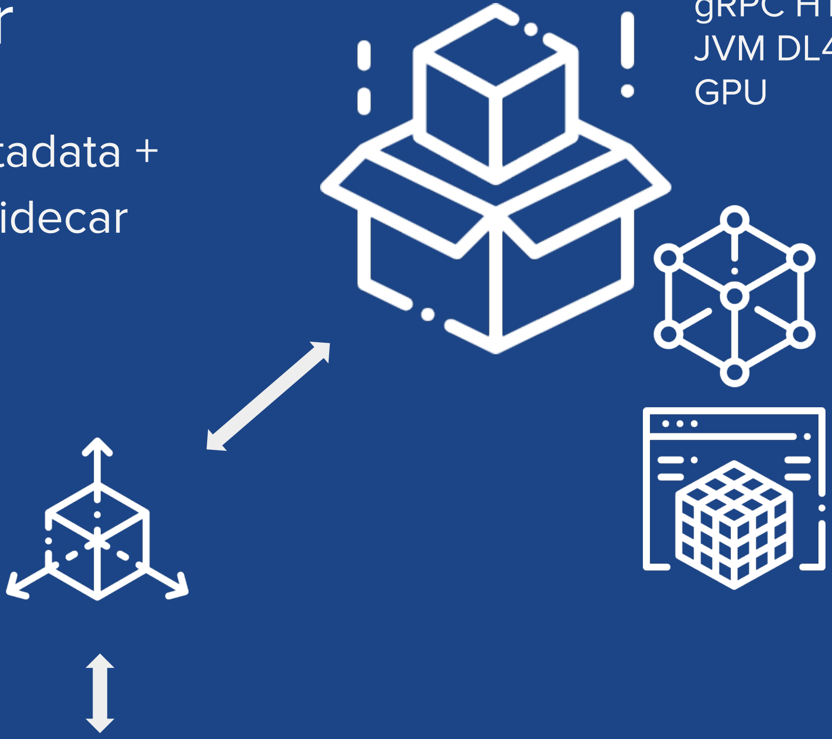
model v2
[...]

routing, shadowing,
pipelining, tracing,
metrics, autoscaling,
A/B, canary

serving requests

Model server

Model artifact + Metadata + Runtime + Deps + Sidecar



gRPC HTTP server
JVM DL4j
GPU

model v2
[...]

predict/
input: bytes image
output: string summary

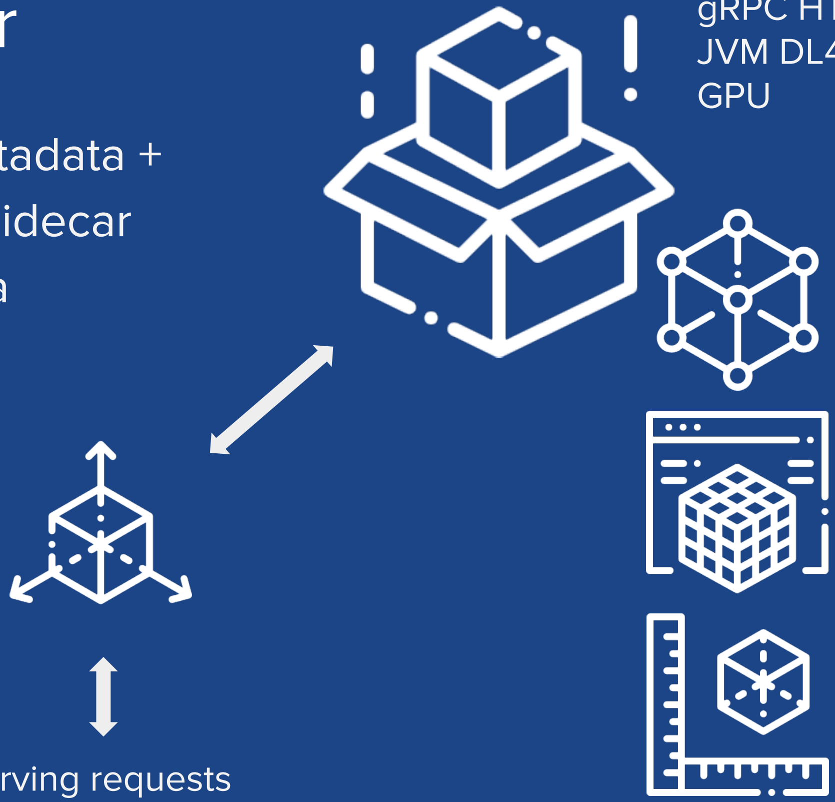
routing, shadowing,
pipelining, tracing,
metrics, autoscaling,
A/B, canary

serving requests

Model server

hydrosphere.io

Model artifact + Metadata + Runtime + Deps + Sidecar + Training metadata



gRPC HTTP server
JVM DL4j
GPU

model v2
[...]

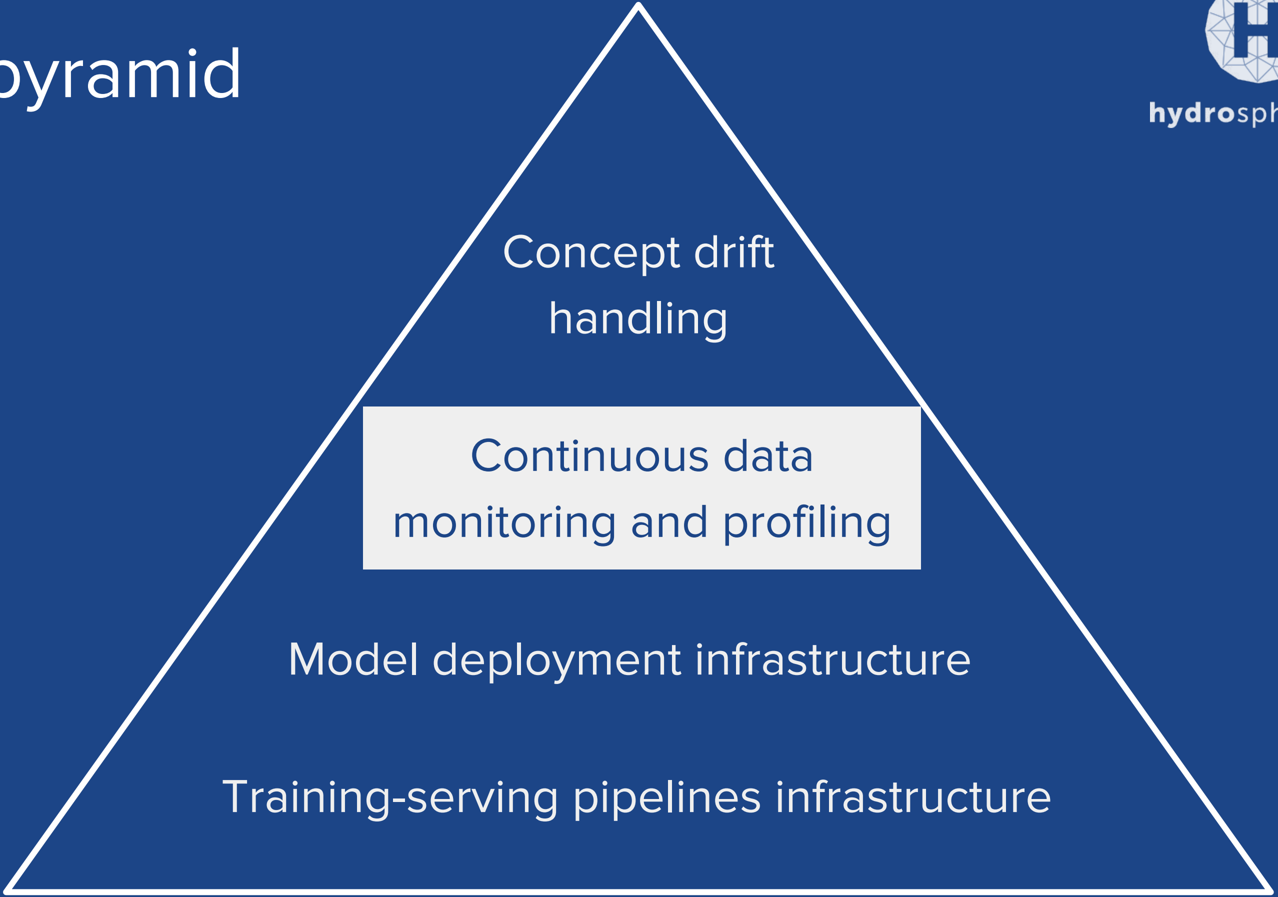
predict/
input: bytes image
output: string summary

min, max
clusters, quantile
autoencoder

routing, shadowing,
pipelining, tracing,
metrics, autoscaling,
A/B, canary

serving requests

AI reliability pyramid



Concept drift handling

Continuous data monitoring and profiling

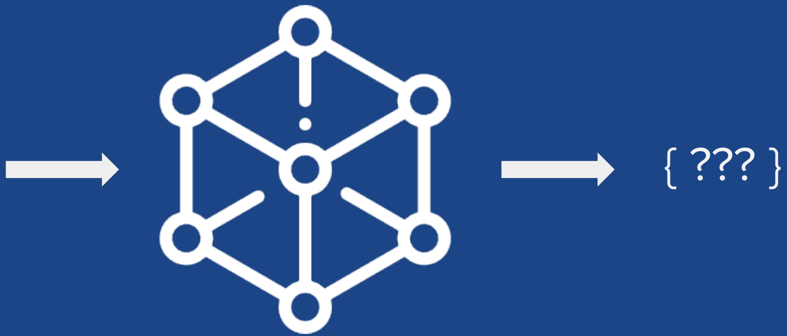
Model deployment infrastructure

Training-serving pipelines infrastructure

hydrosphere.io

Data format drift

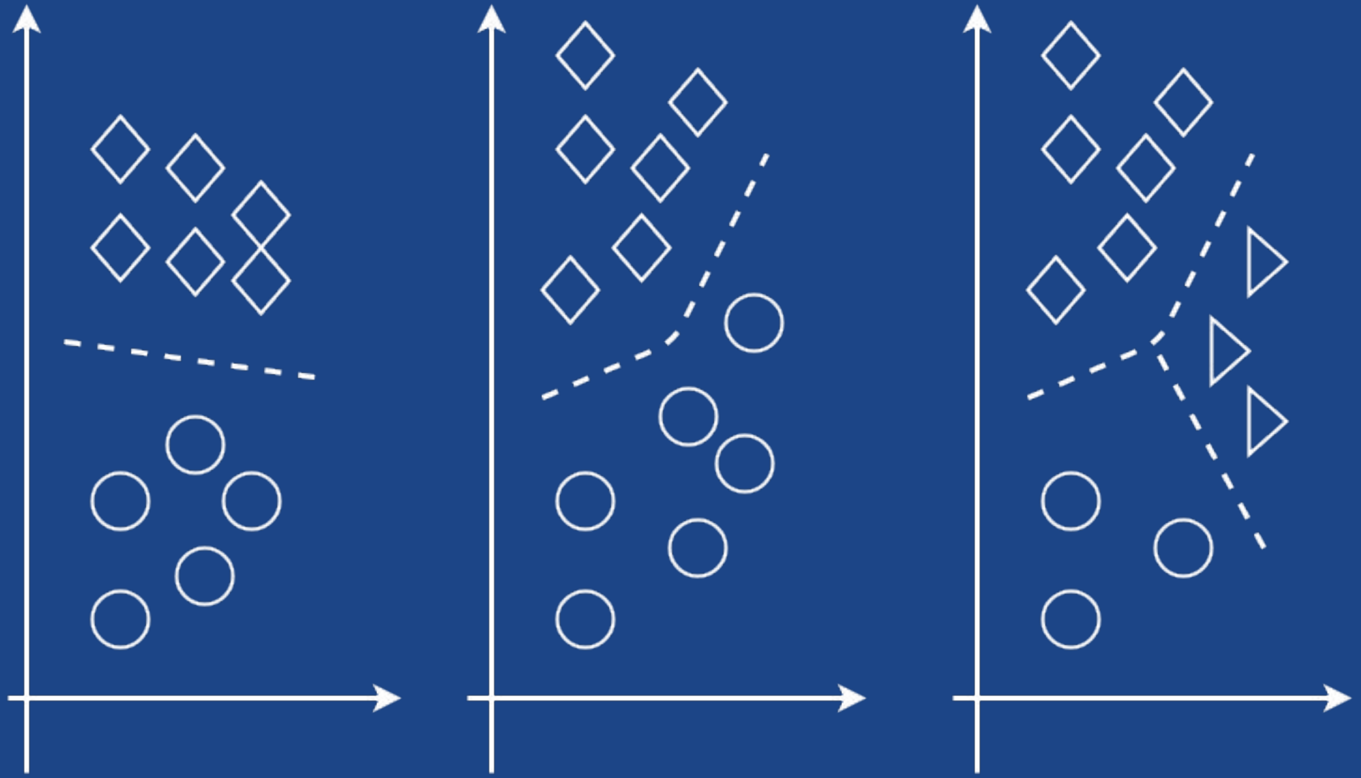
{ age: 30, 25, …, 1986, “1990” }
{ wage: 150, 150000, …, “10k”, “12.000”}
{ gender: “male”, “female”, “man”, 1 }



{ ??? }

Concept drift / original / drifted / new

hydrosphere.io



Data `exploration` in production

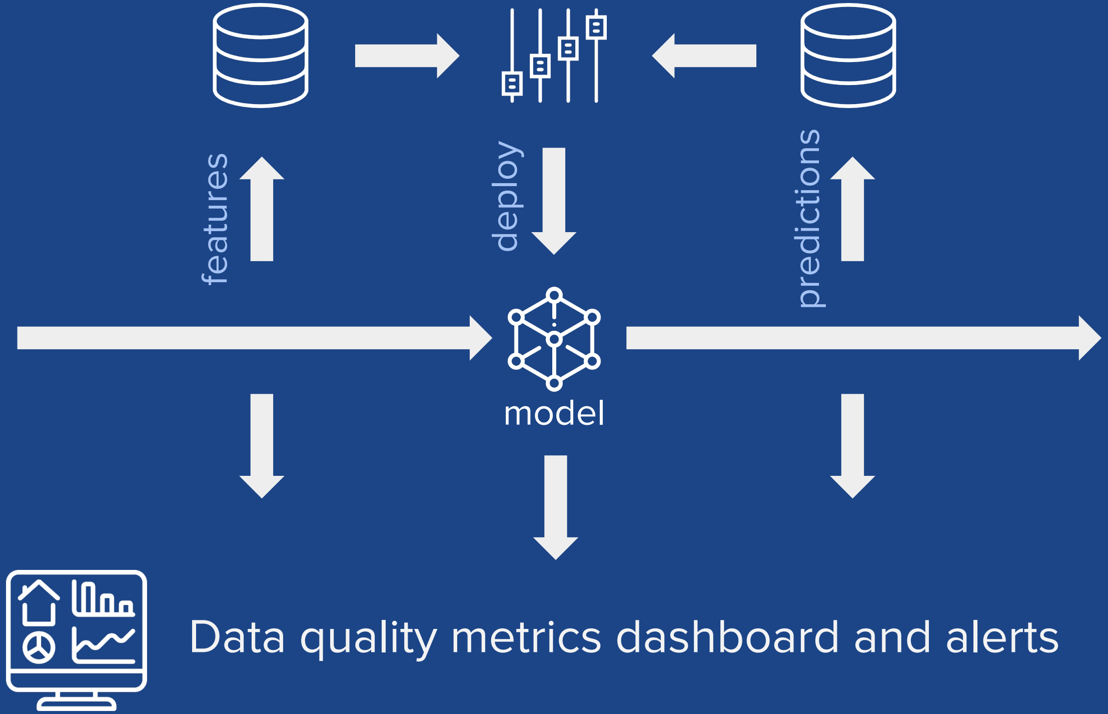
Research: Data scientist makes assumptions based on results of data exploration →

Production: The model works iff format and statistical properties of data are the same as in research

Data exploration by scientist

Continuous data exploration and validation

Metrics



Data quality metrics dashboard and alerts

How to deal with … ?

hydrosphere.io

- multidimensional dataset
- data timeliness
- data completeness
- text, image data
- complicated seasonality

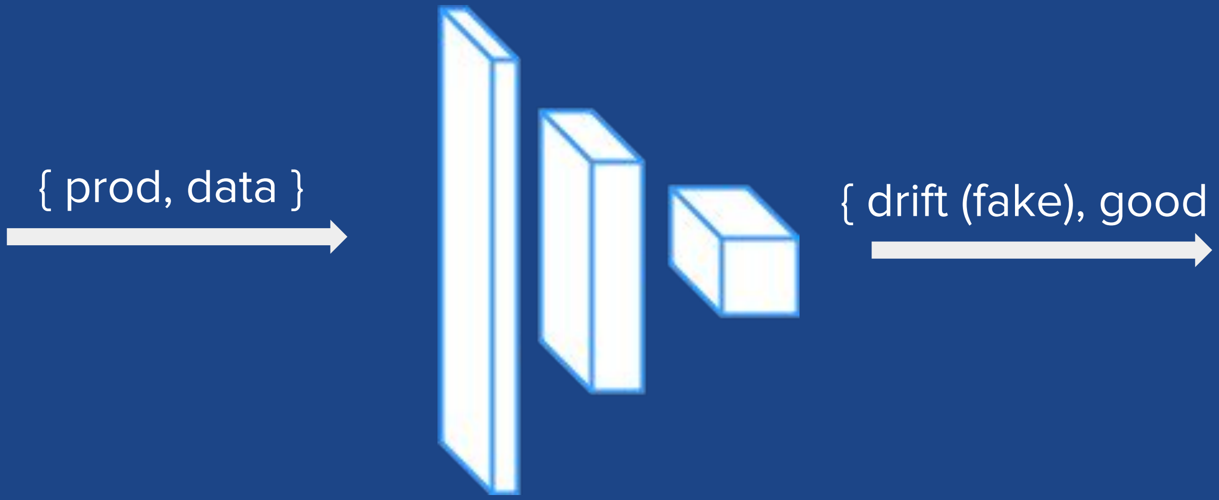
Send a maniac to catch a maniac

© Warner Bros. 1993

Metrics

- Kolmogorov-Smirnov test
- Q-Q plot, t-digest
- Spearman and Pearson correlations
- Density based clustering algorithms
- Deep Autoencoders
- Generative Adversarial Networks
- MADE - Masked Autoencoder Density Estimation
- Random Cut Forest
- Model specific metrics

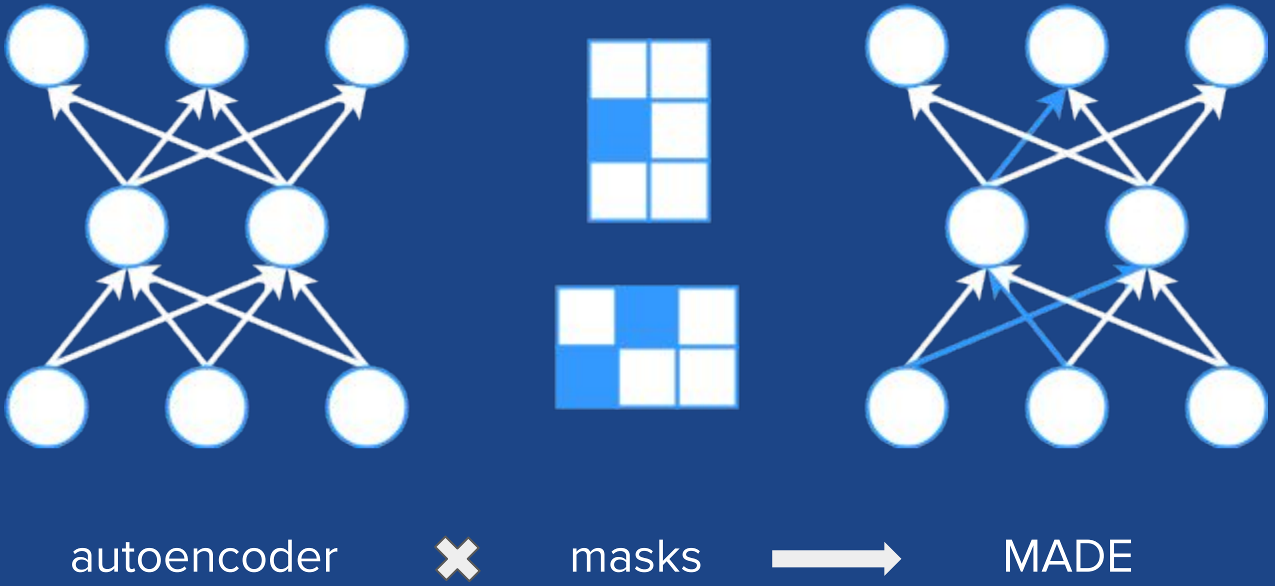
GAN / Discriminator



{ prod, data }

{ drift (fake), good }

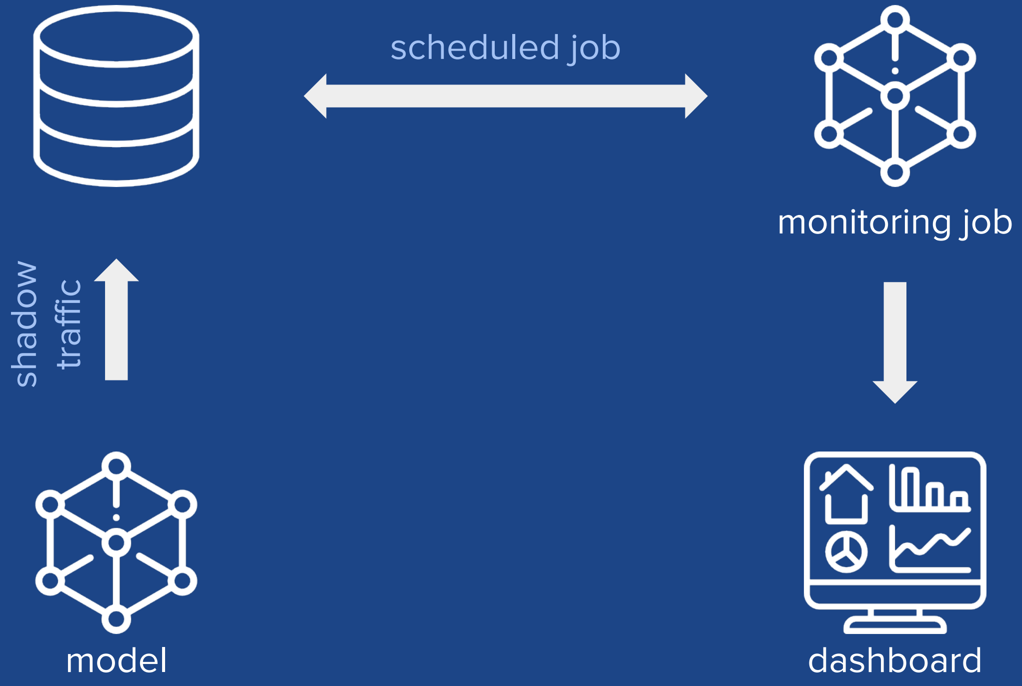
MADE



autoencoder × masks → MADE

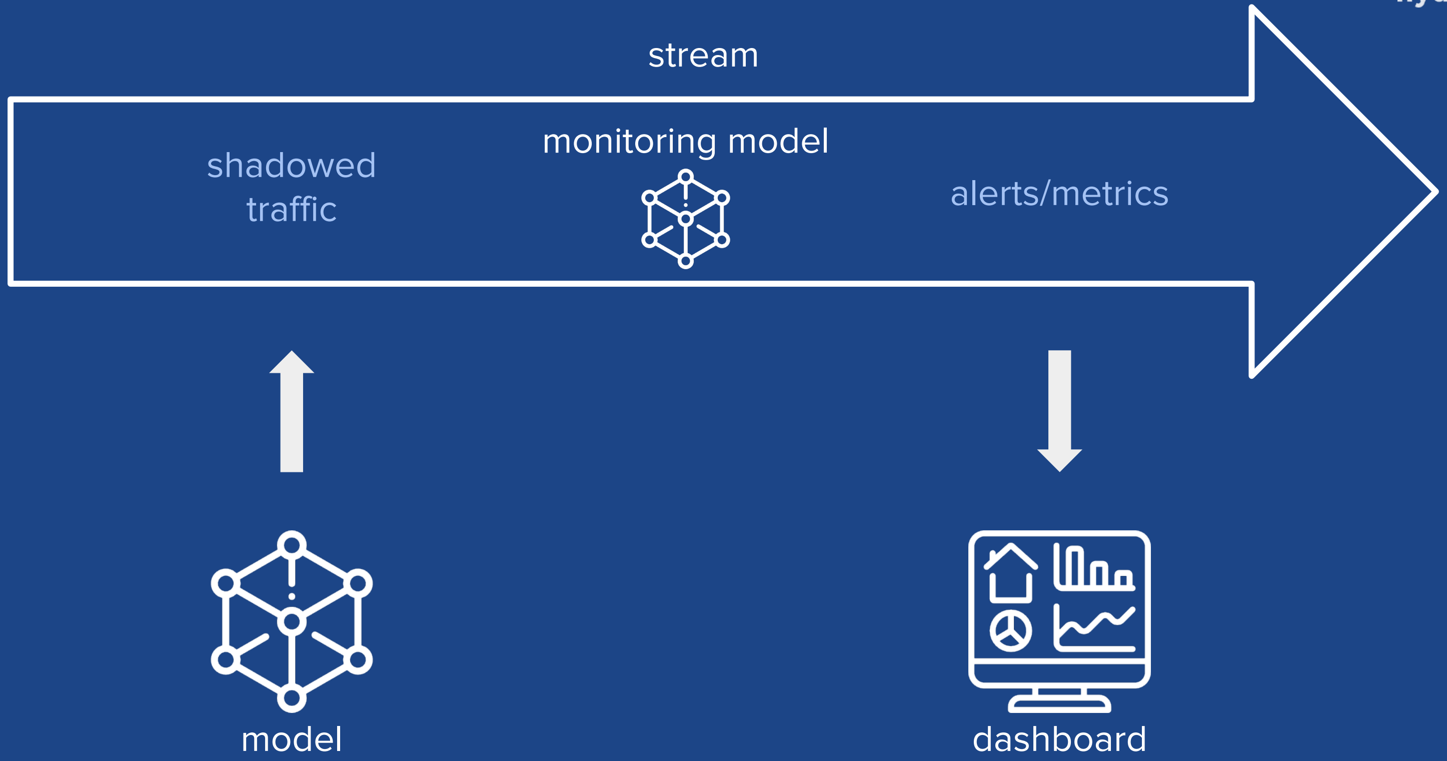
Deployment and monitoring / ETL

hydrosphere.io



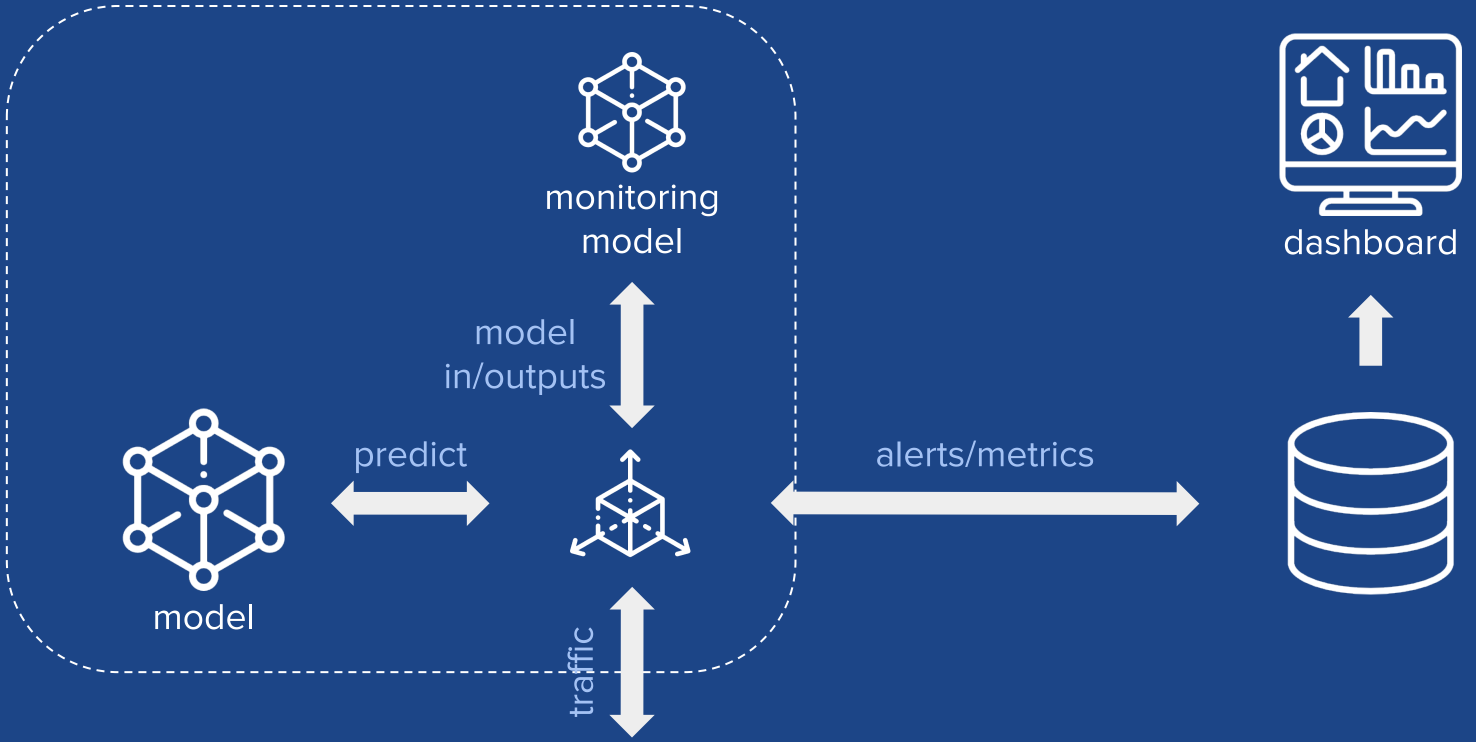
Deployment and monitoring / Streaming

hydrosphere.io

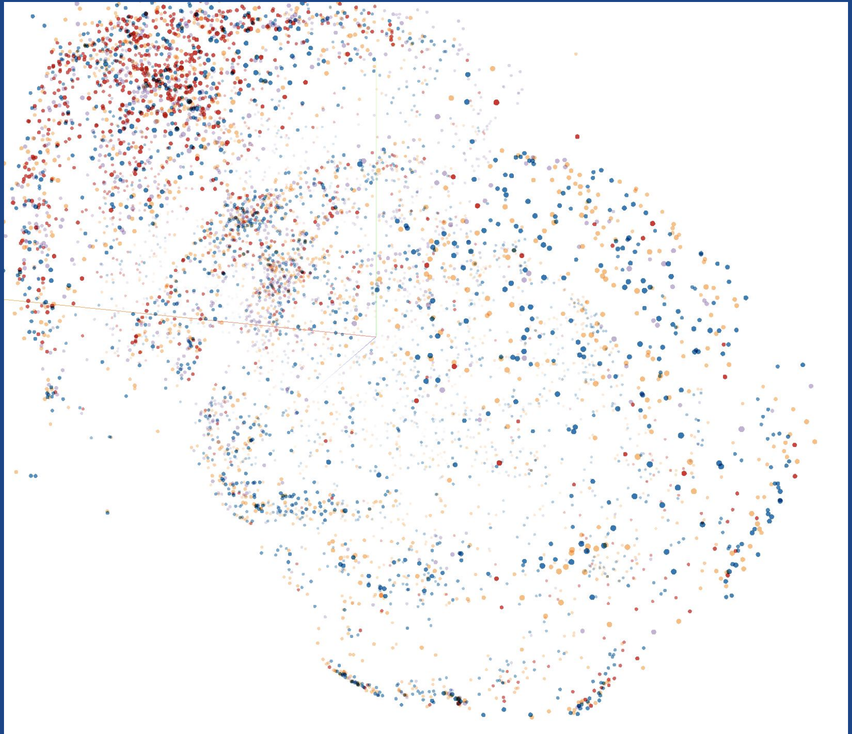


Deployment and monitoring / Sidecar

hydrosphere.io



Example / NL systems

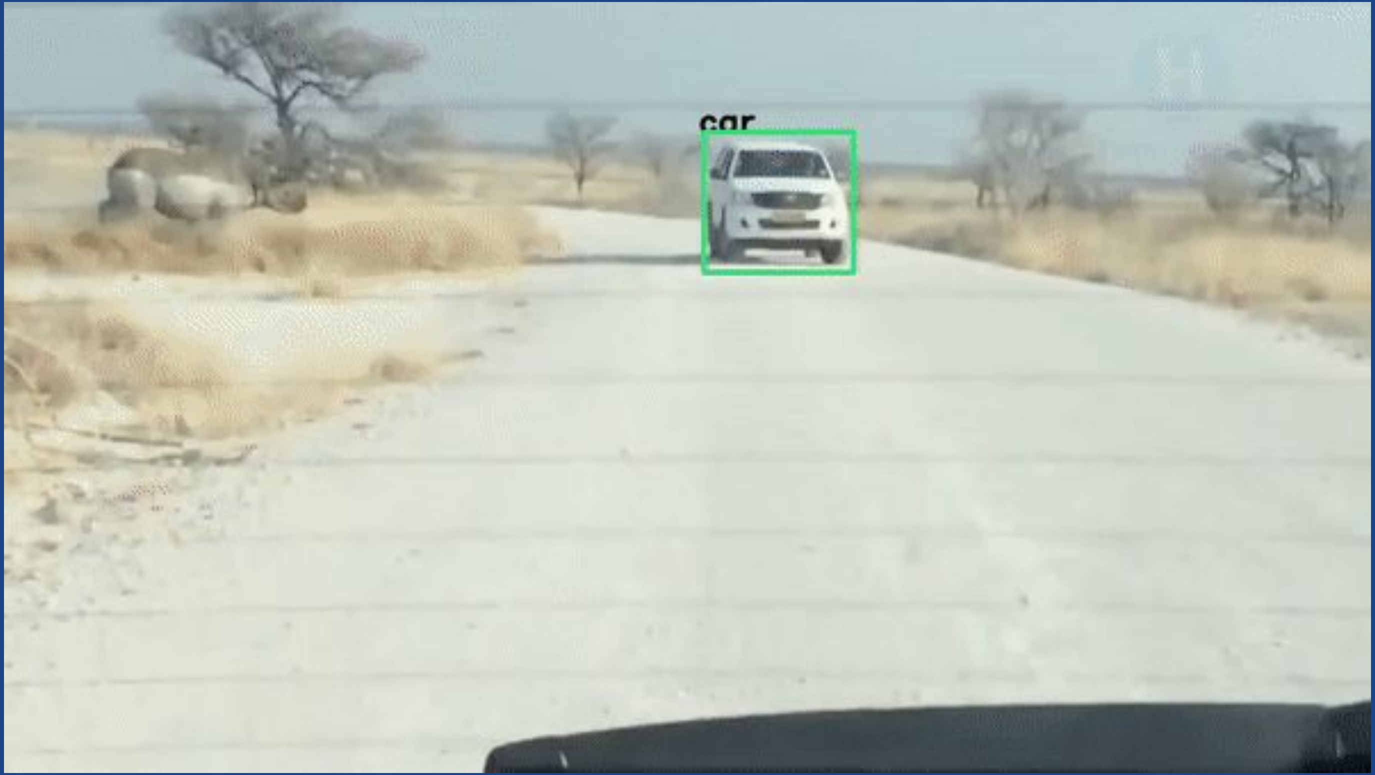


Red and Purple - cluster of “bad” production data

Yellow and Blue - dev and test data

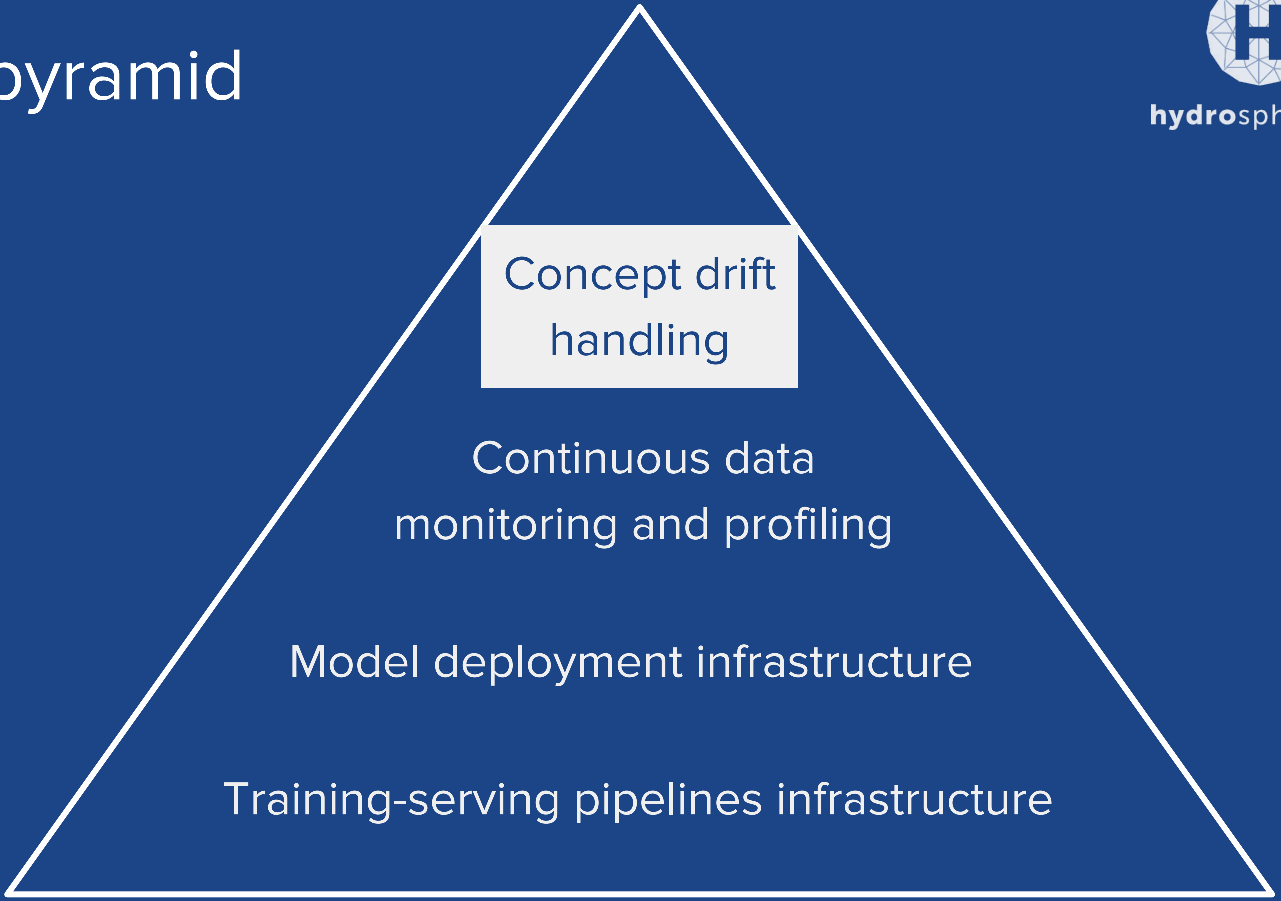
Example / Images

hydrosphere.io



AI reliability pyramid

hydrosphere.io


Concept drift handling

Continuous data monitoring and profiling

Model deployment infrastructure

Training-serving pipelines infrastructure

Model retraining

hydrosphere.io

When to retrain? When/how to push to prod? What data to retraining with?

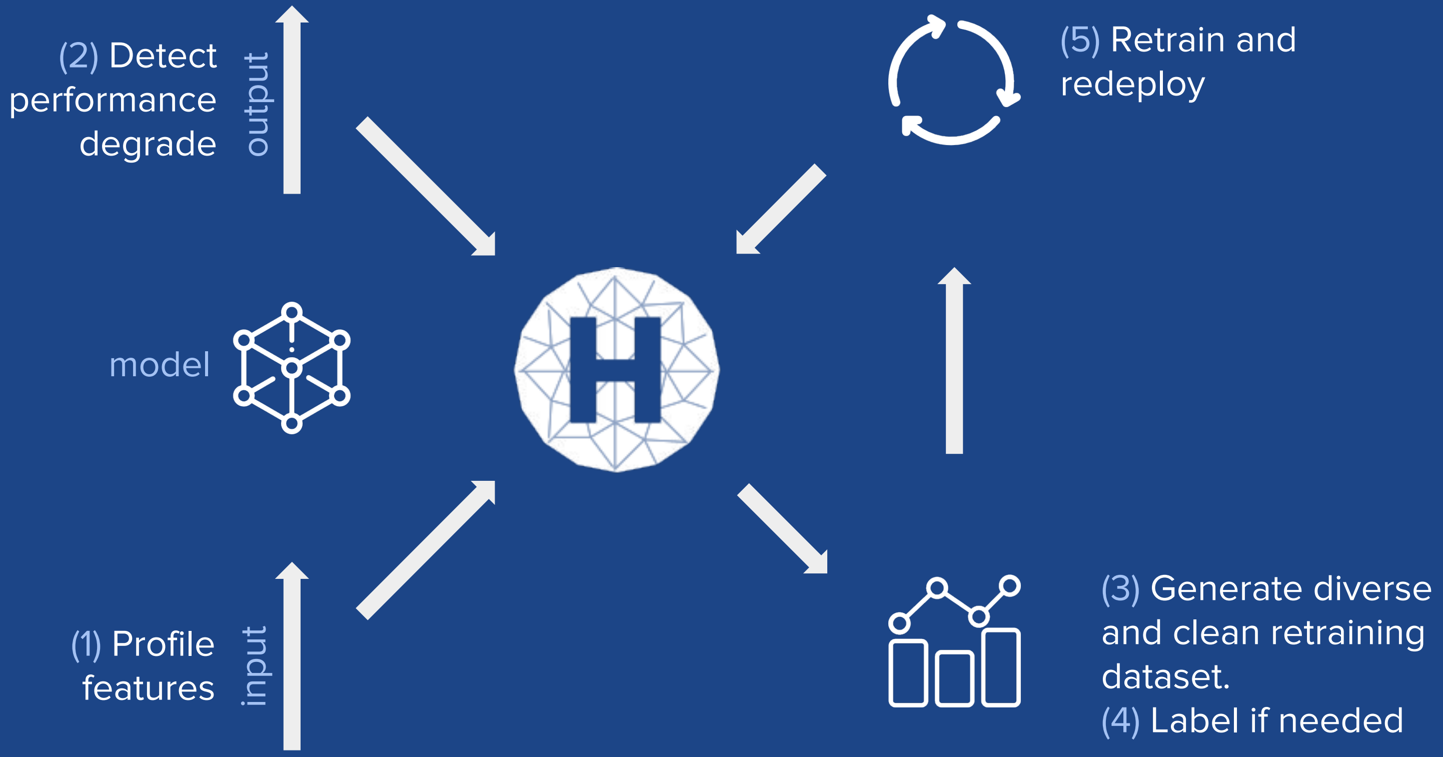
Manually

- Works well for 1 model
- Does not scale

Automatically with the latest batch

- Not safe
- Can be expensive
- The latest batch may not be representative

Solution / Reactive AI powered retraining



(2) Detect performance degrade

output

model

(1) Profile features

input

(5) Retrain and redeploy

(3) Generate diverse and clean retraining dataset.

(4) Label if needed

Thank you!

hydrosphere.io

Iskandar Sitdikov
isitdikov@hydrosphere.io