

Machine Learning on Flink



Ted Dunning
MapR Technologies

Machine Learning on Flink (without Flink ML)

The MapR logo is displayed in white text on a red rectangular background. The logo consists of the word "MAPR" in a bold, sans-serif font, with a registered trademark symbol (®) to the upper right of the "R".

MAPR®

Ted Dunning
MapR Technologies

Machine Learning on Flink (without Flink ML*)

The MapR logo is displayed in white text on a red rectangular background. The logo consists of the word "MAPR" in a stylized, bold, sans-serif font, followed by a registered trademark symbol (®).

MAPR®

Ted Dunning
MapR Technologies

Machine Learning on Flink (without Flink ML*)

*well, with Flink ML if desired, but not just with Flink ML

Ted Dunning
MapR Technologies

The MapR logo is displayed in white text on a red rectangular background. The logo consists of the word "MAPR" in a bold, sans-serif font, with a registered trademark symbol (®) to the upper right of the letter "R".

MAPR®

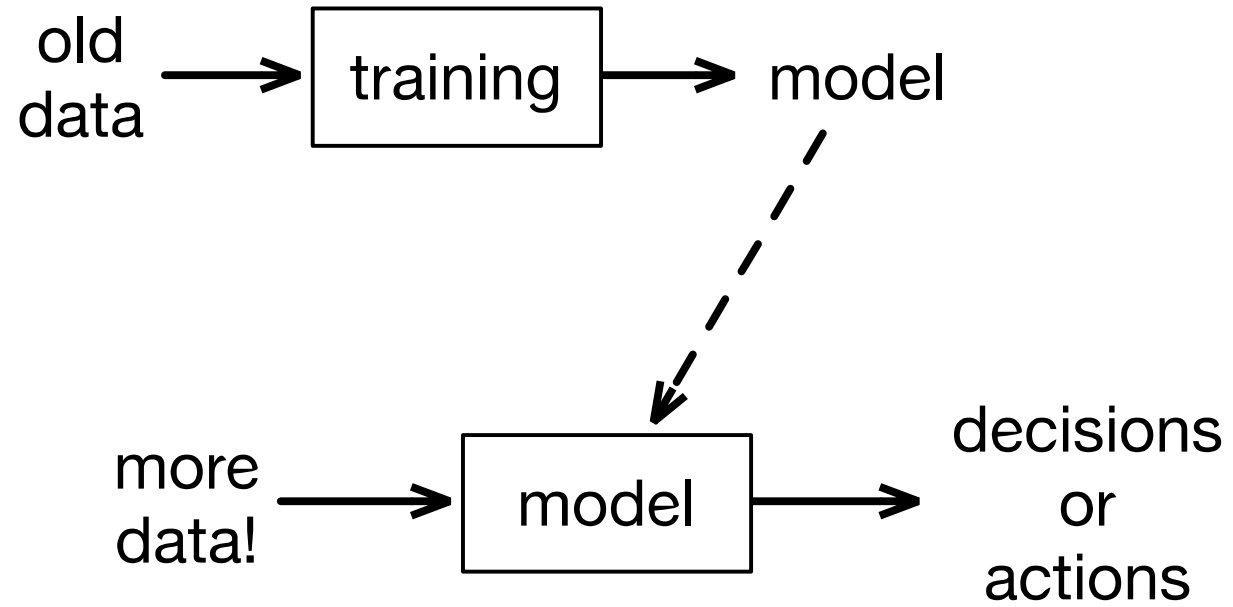
Where to Today?

- What is important in machine learning systems?
- What isn't like it seems?
- What can I/we/you do to improve it?

Traditional View



Traditional View



Not so much!

90+% of effort is logistics,
not learning

Why Not?

- Just getting the training data is hard
 - Bootstrap problem for first models
 - First models and later can record input data
 - New kinds of observations force restarts
 - Requires a ton of domain knowledge
- The myth of the unitary model
 - You can't train just one
 - You will have dozens of models, likely hundreds to thousands
 - Handoff to new versions is tricky

Why Not? (continued)

- Recording raw-ish data is really a **big** deal
 - Data as seen by a model is worth gold
 - Data reconstructed later often has time-machine leaks
 - Databases were made for updates, streams are safer
- Real validation can be hard and takes time
 - Especially true if models effectively pick their own training data
 - Consider model over-ride for interesting data
 - Reject inference can be very difficult without a good framework
 - Thompson sampling can provide (near) optimal solution

How to Do Better ... Spoilers

- It's the data!
- Prioritize – put serious effort into infrastructure
- Persist – use streams to keep data around
- Measure – everything, and record it
- Meta-analyze – understand and see what is happening
- Containerize – make deployment repeatable, easy
- Oh... don't forget to do some machine learning, too

How to Do Better – Big Picture

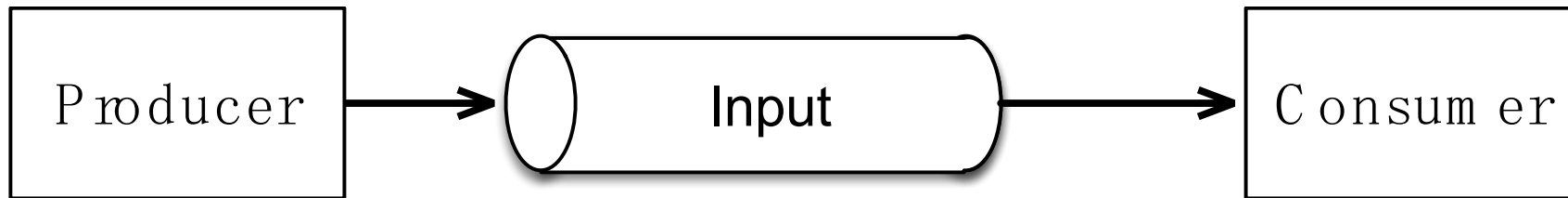
- The biggest improvements probably aren't machine learning
 - Algorithms are fun, but probably not the biggest bang
- Acquiring better data can make massive difference
 - Example: Clicks versus 30 second views
- Asking a better question can make massive difference
 - Example: When does new stock arrive versus stock recommender
- The 17th model revision will probably make <1% difference
- Consider \$/minute of effort, invest wisely

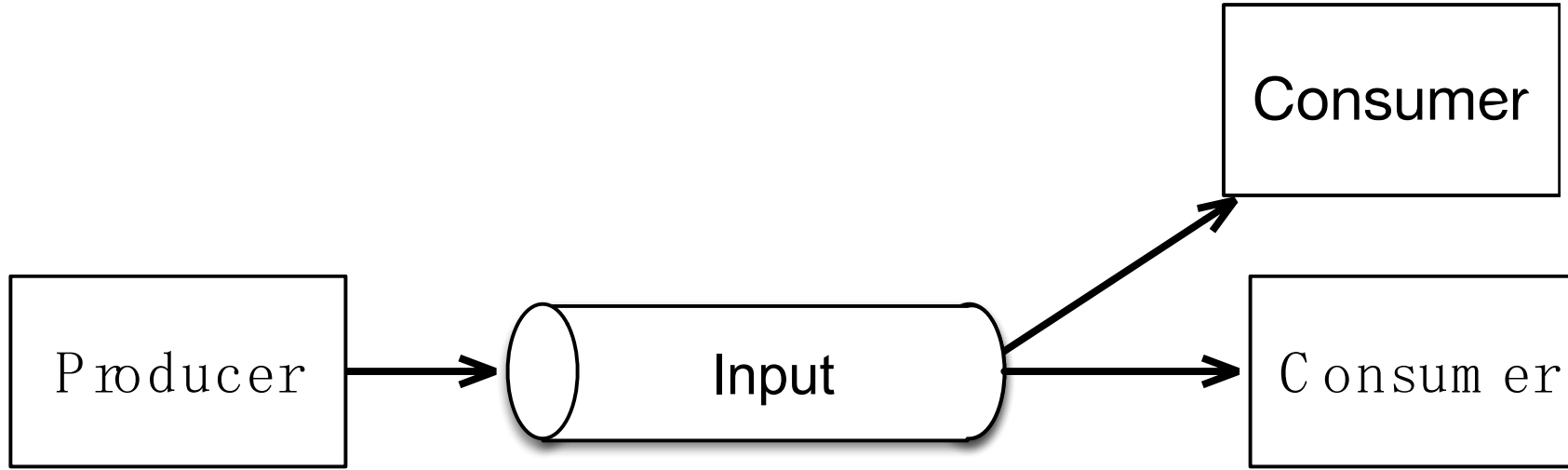
How to Do Better – Data and Deployment

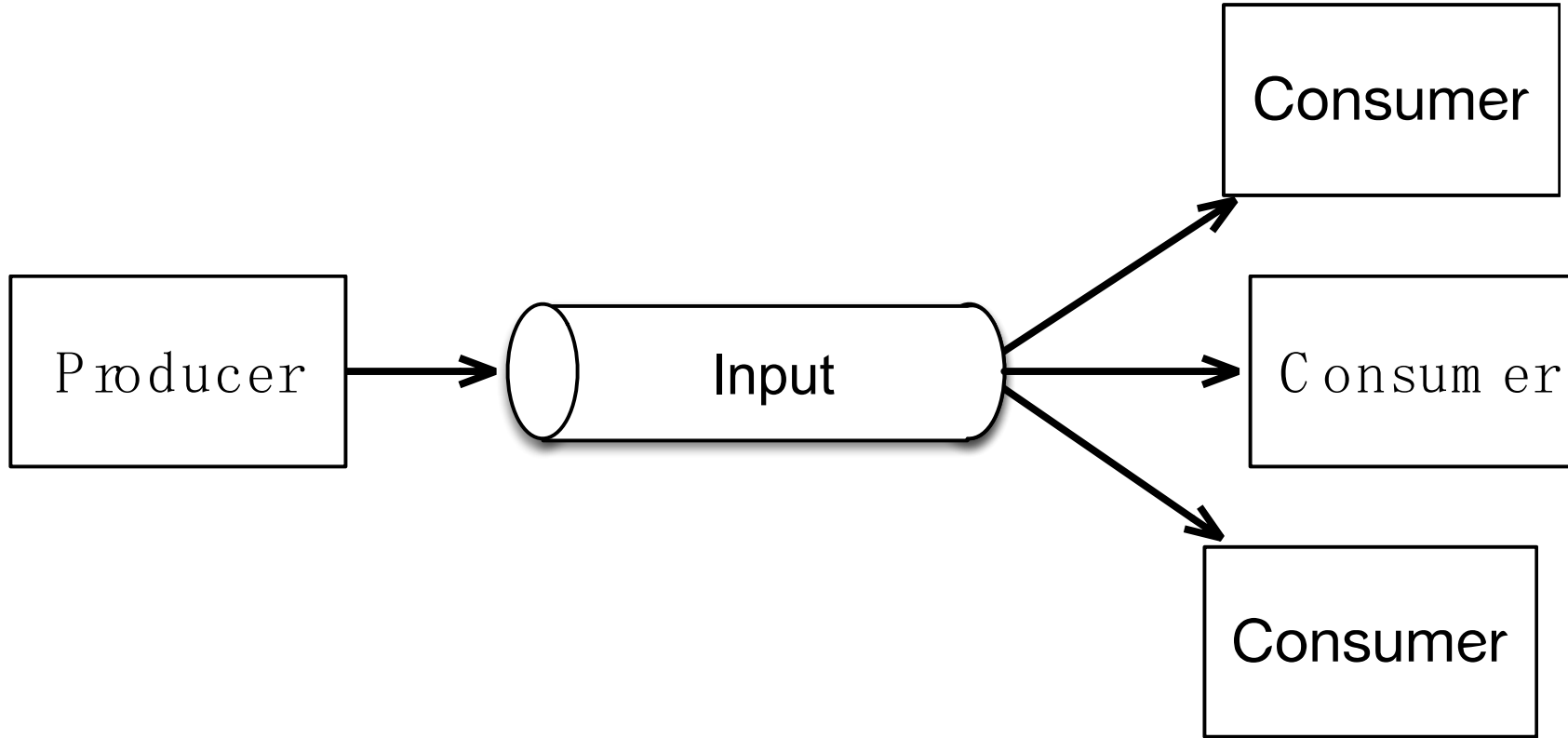
- Deploy a decoy model
 - Looks like a server, but it just archives inputs
 - Safe in a good streaming environment, less safe without good isolation
- Deploy a canary server
 - Keep an old model active as a reference
 - If it was 90% correct, difference with any better model should be small
 - Score distribution should be roughly constant
 - Many things can disturb the canary, but if it is stable, things are likely OK

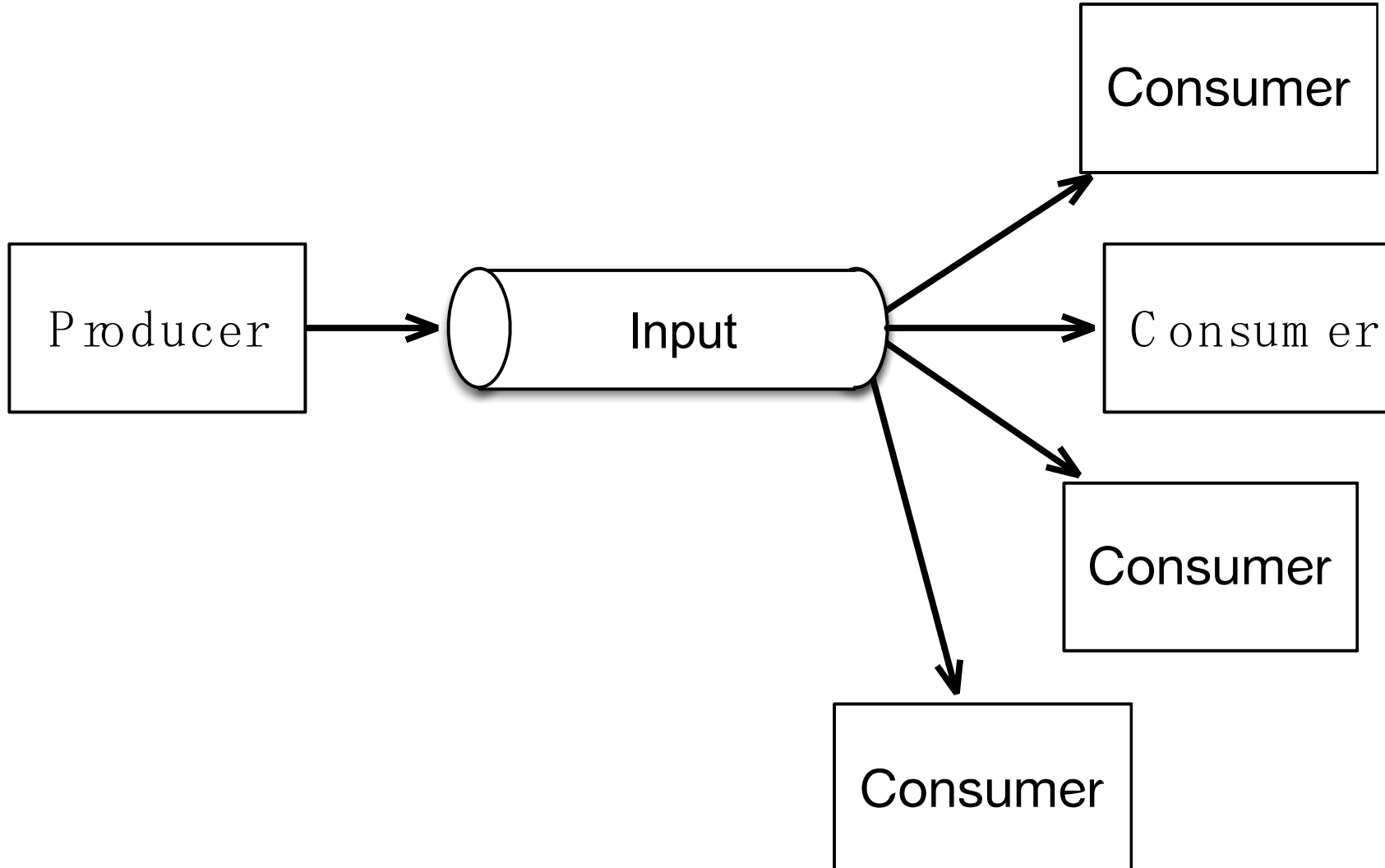
A decoy and a canary walked into a bar
and asked for a bottle of Club Maté..

Bartender said, “Wouldn’t
you rather have a stream?”







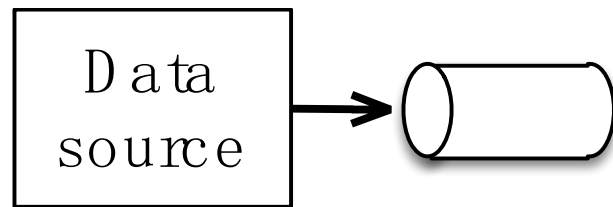


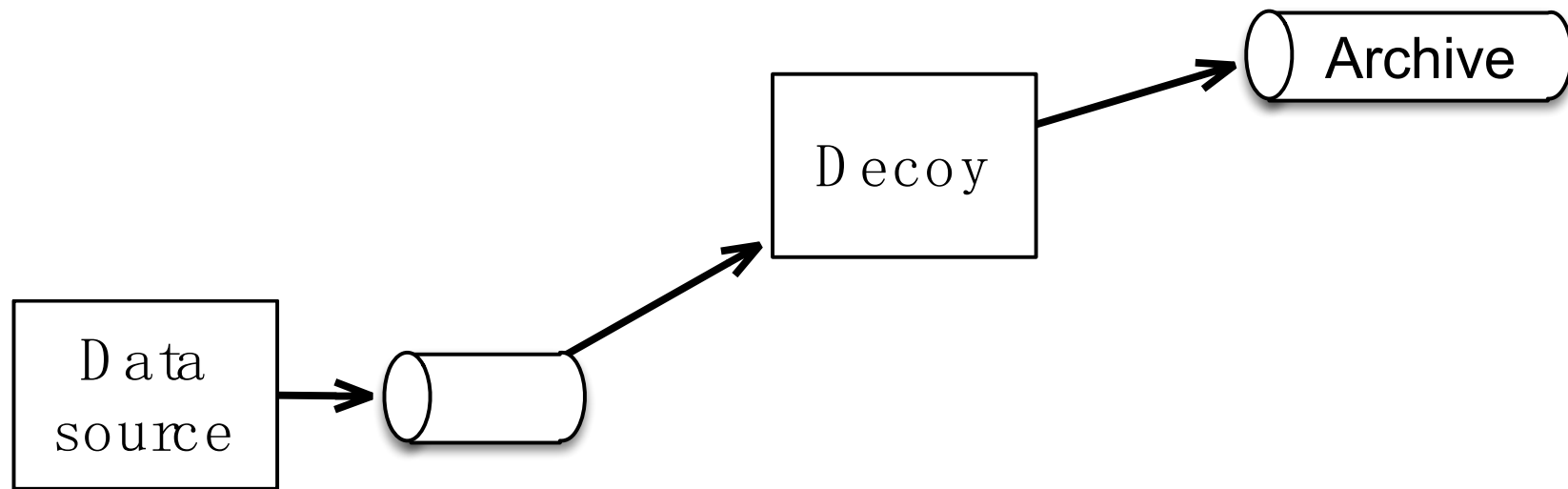
Features of Good Streaming

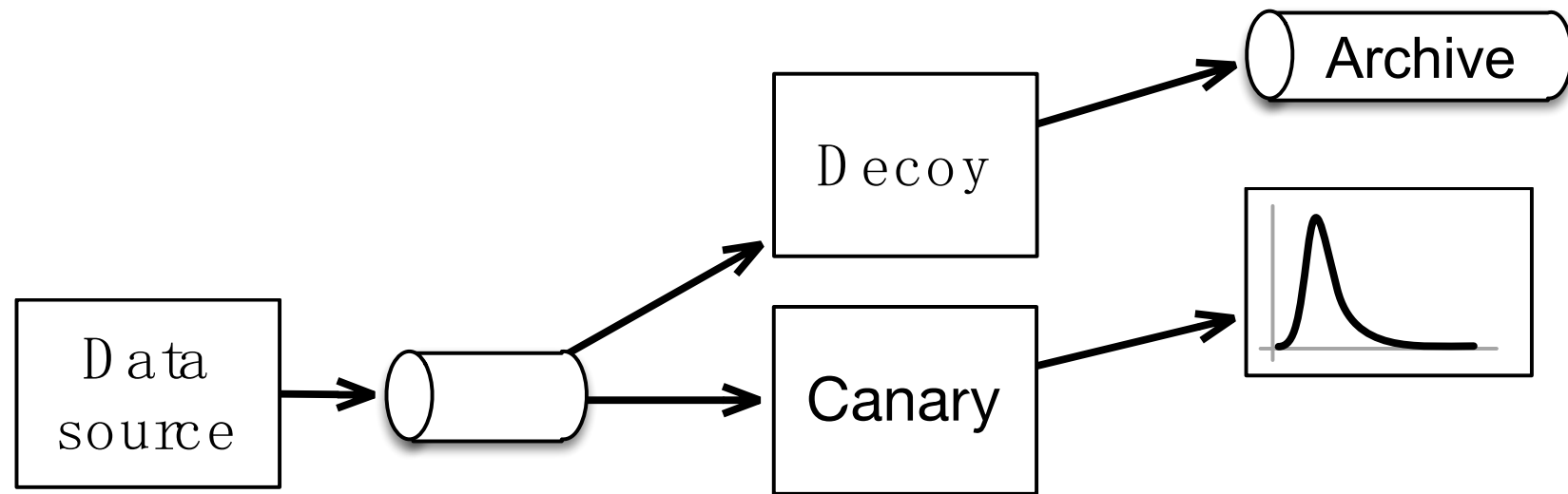
- It is Persistent
 - Messages stick around for other consumers
 - Consumers don't affect producers
- It is Performant
 - You don't have to worry if a stream can keep up
- It is Pervasive
 - It is there whenever you need it, no need to deploy anything
 - How much work is it to create a new file? Why harder for a stream?

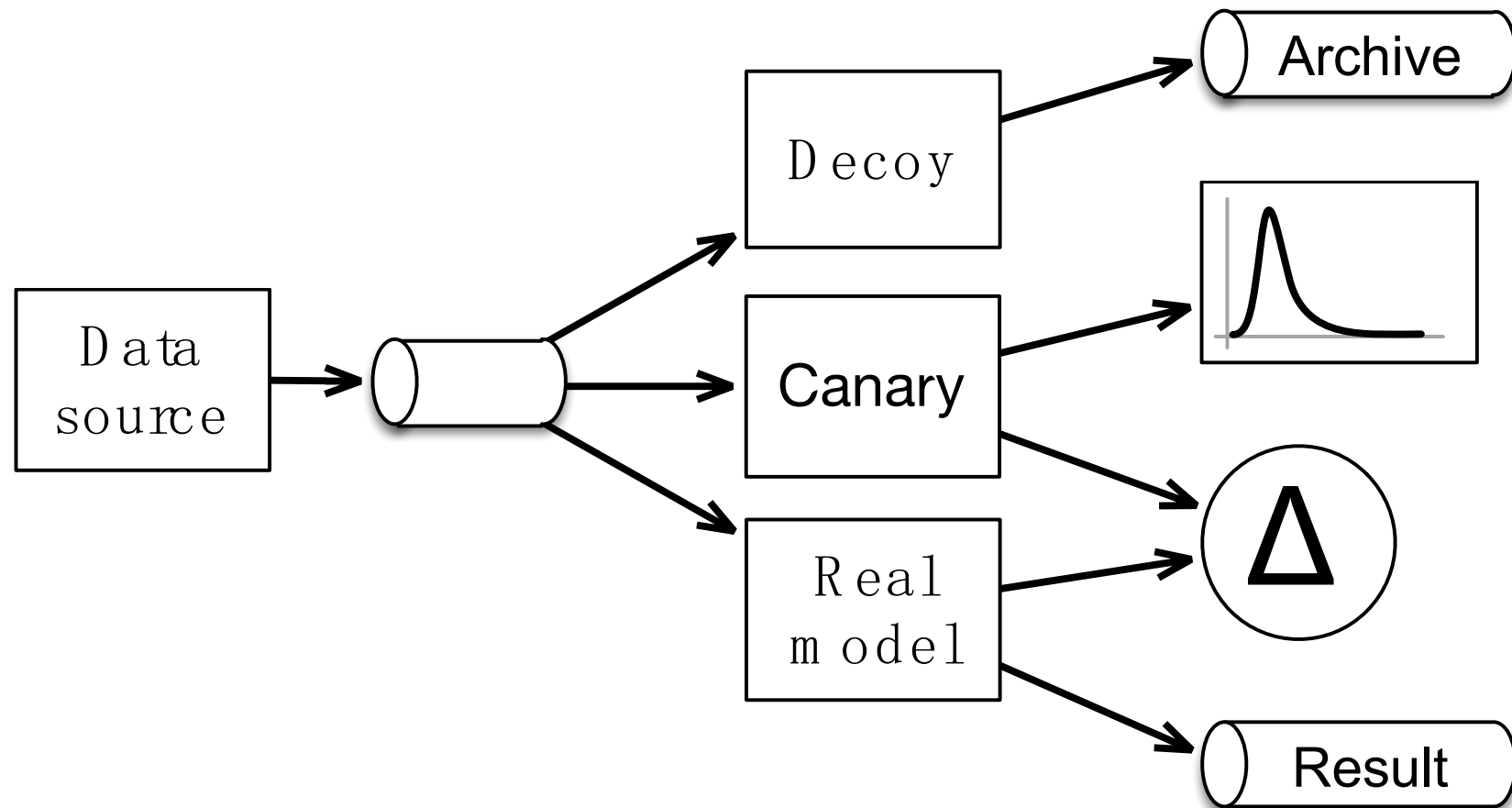
How to Do Better – Data and Deployment

- Deploy a decoy model
 - Looks like a server, but it just archives inputs
 - Safe in a good streaming environment, less safe without good isolation
- Deploy a canary server
 - Keep an old model active as a reference
 - If it was 90% correct, difference with any better model should be small
 - Score distribution should be roughly constant
 - Many things can disturb the canary, but if it is stable, things are likely OK



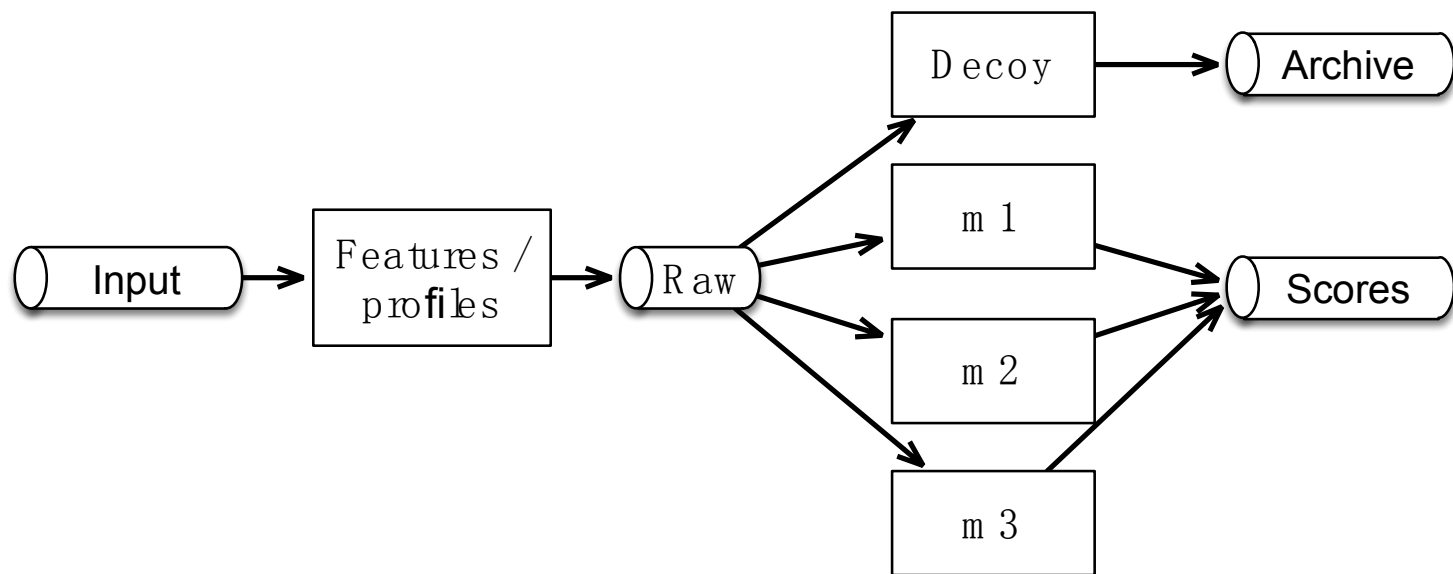






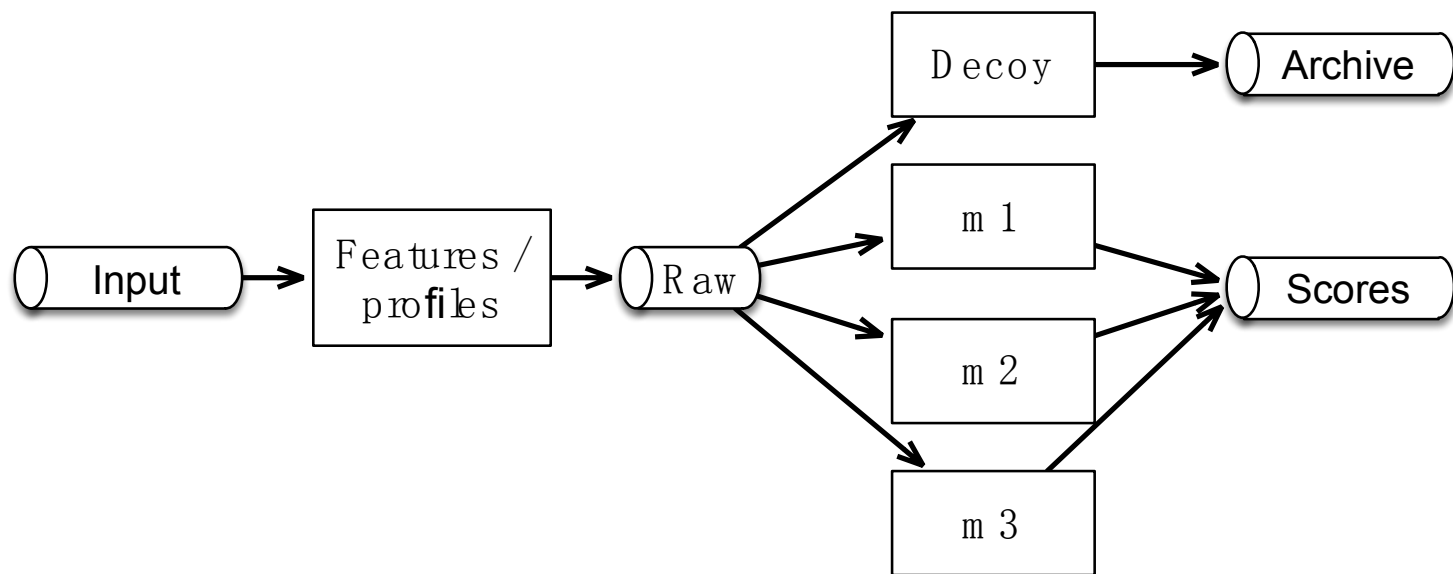
How to Do Better – Containers and Rendezvous

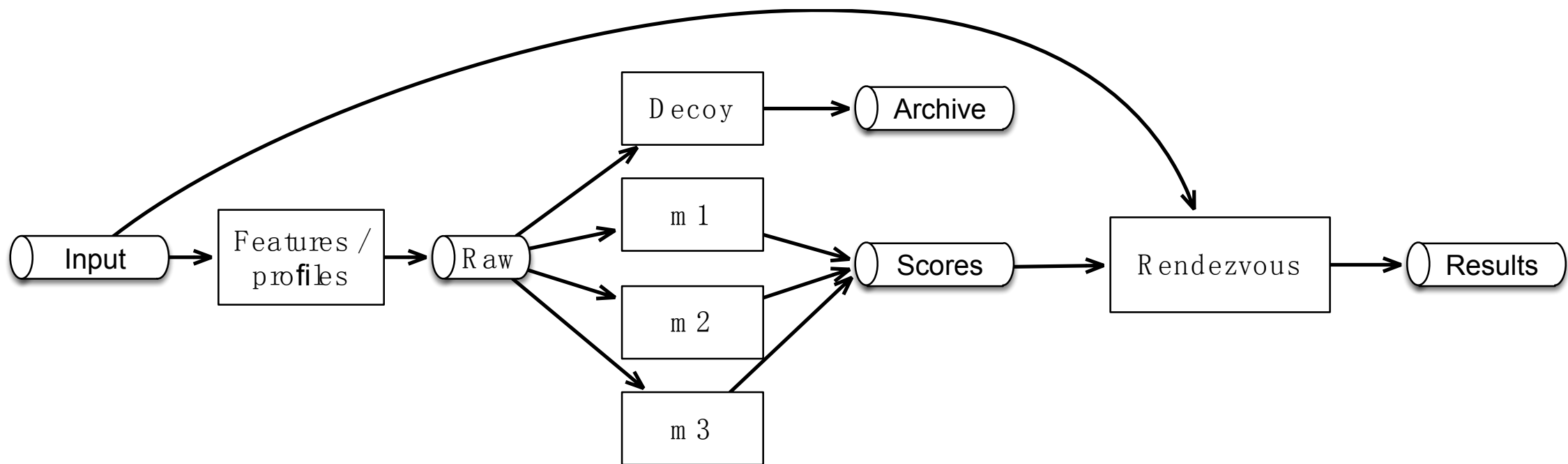
- Containerize all models
 - Safe deployment
 - Take input from stream
 - Write output to stream
 - Annotate output with meta-data about who, what and when
 - Pre-rig for standard operations like metrics, timing, annotation

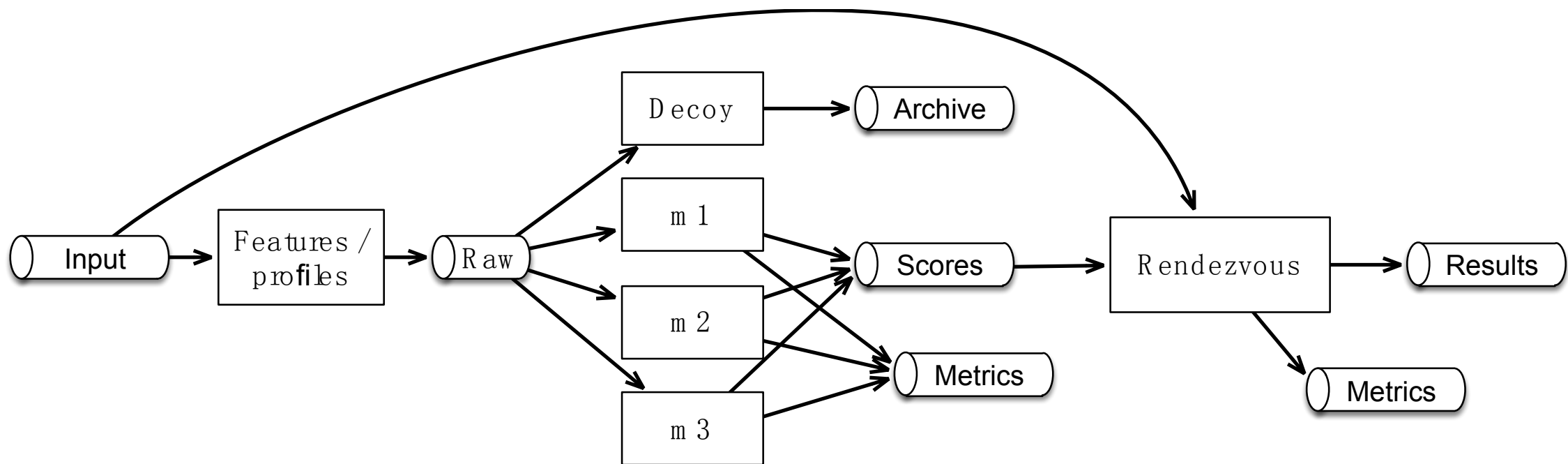


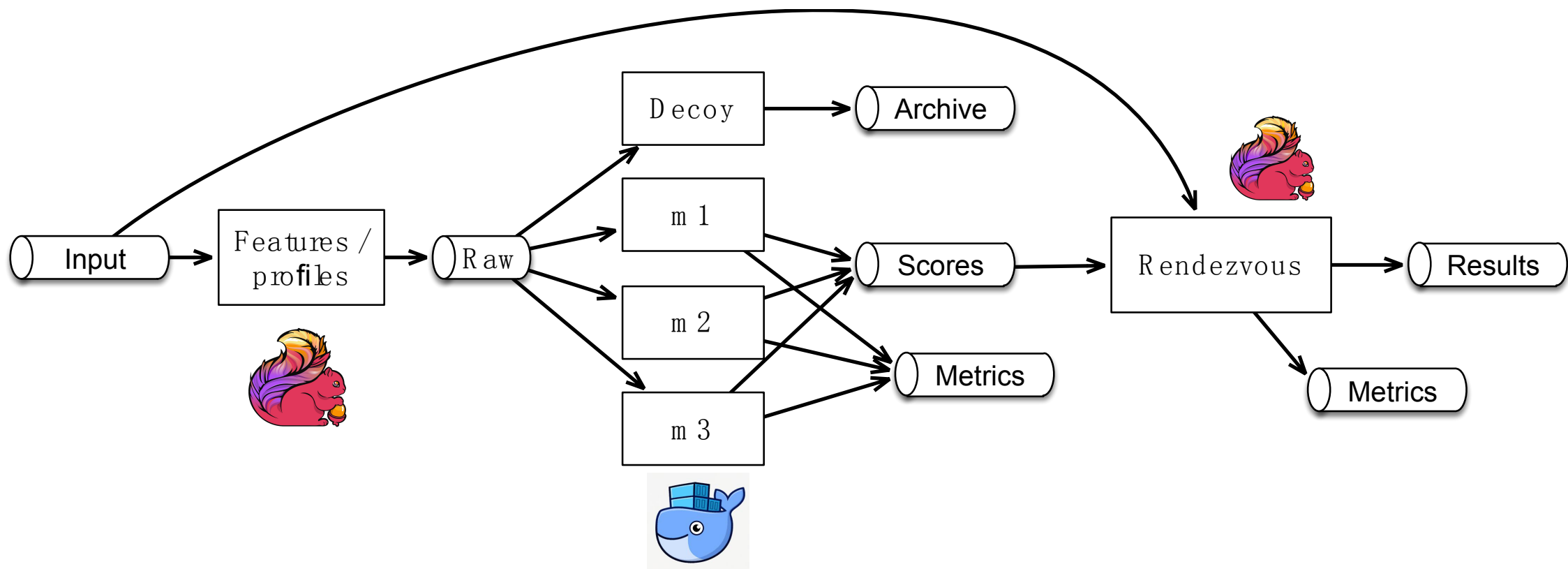
How to Do Better – Containers and Rendezvous

- Select desired model result
 - Thompson sampling, dithering
 - Defaults after extraordinary delays
- Rendezvous with original input
 - Or carry origin meta-data
- Record metrics
 - Latency
 - Score distributions
 - Selected model







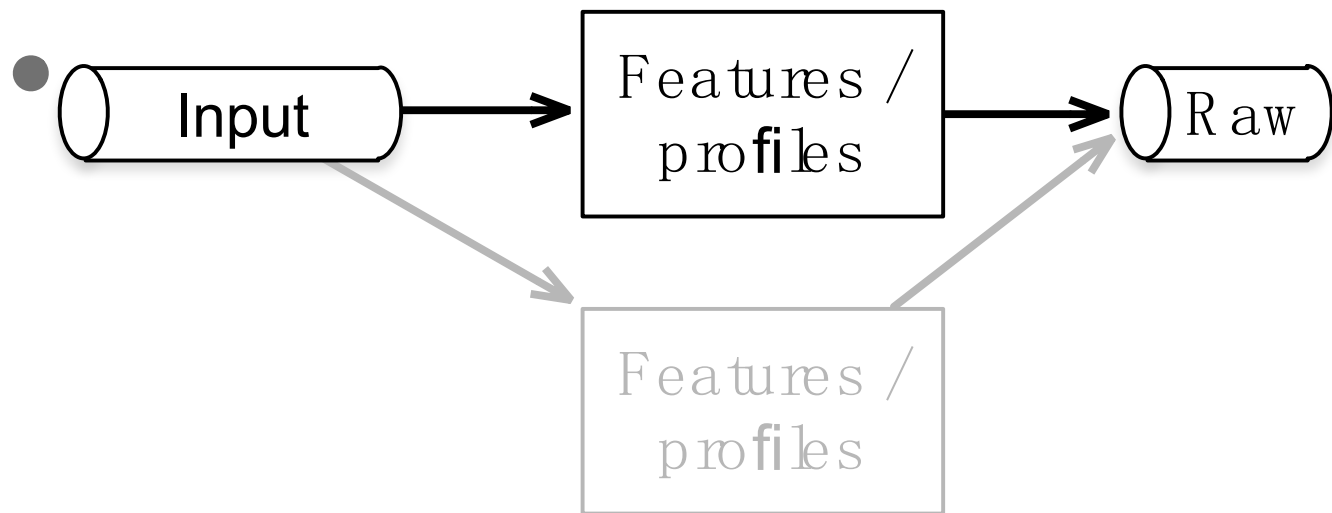


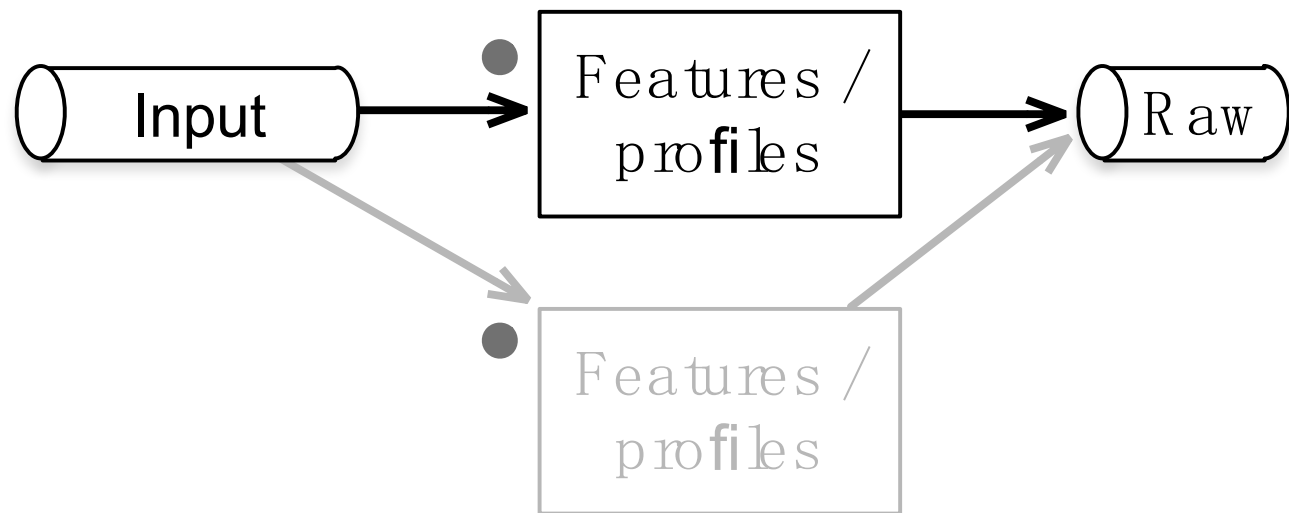
How to Do Better – Model Deployment as a Data Op

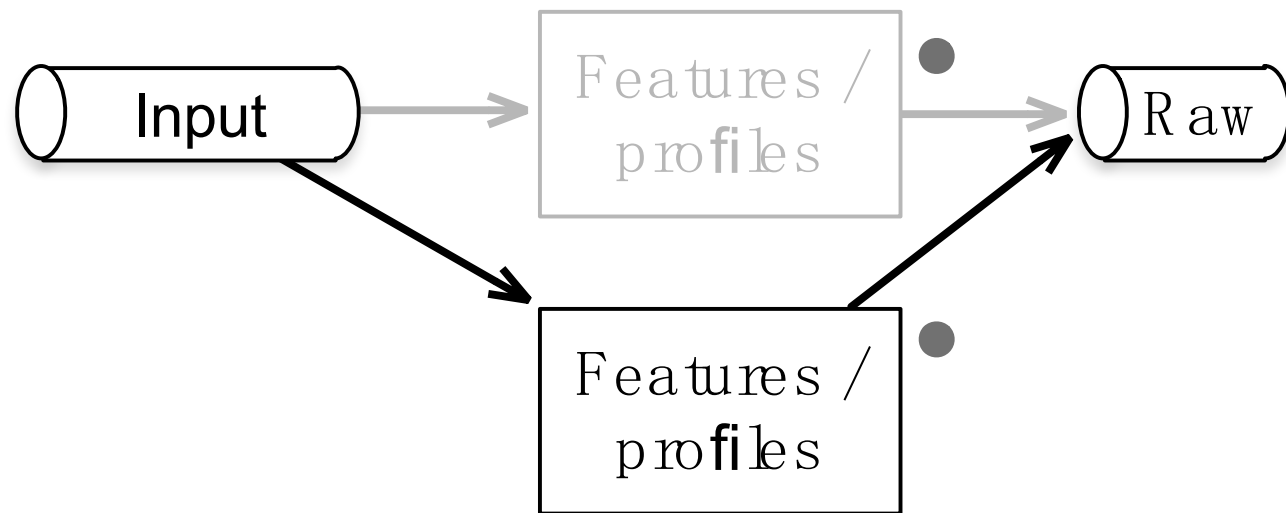
- Deploying a new model means spin some new containers
- Results will flow into the scoring stream
 - And will be ignored (initially)
- Tell the rendezvous service to use new model results
 - Start slowly
 - Check latencies, delta versus champion and canary
- Rendezvous service needs fallback rules
 - Use m3, after x ms use m2 if available, after y ms more, use m1
 - Must ***always*** give some answer within the SLA
 - Fancy statistics possible

Who Deploys the Deployer?

- Flink savepoints can be used to deploy new feature extractor, rendezvous service
- Underlying Chandry-Lamport algorithm can also be coded explicitly (not recommended for most folks)
- Hot swap token propagates through system to indicate when to switch







Takeaways

- It's the data!
- Prioritize – put serious effort into infrastructure
- Persist – use streams to keep data around
- Measure – everything, and record it
- Meta-analyze – understand and see what is happening
- Containerize – make deployment repeatable, easy
- Oh... don't forget to do some machine learning, too

Resources

- General principles of ML engineering with a Googlish accent:
<http://bit.ly/ml-best-practices-1>
- More good practices:
<http://bit.ly/ml-best-practices-2>
- Best short feature on feature engineering I know about:
<http://bit.ly/feature-engineering-1>

Who I am

Ted Dunning, Chief Applications Architect, MapR Technologies

Board of Directors, Apache Software Foundation

Email tdunning@mapr.com tdunning@apache.org

Twitter @Ted_Dunning

Q&A

Engage with us!

@mapr



maprtech

mapr-technologies



MapR

tdunning@mapr.com



maprtech