Online machine inference and learning in practice

PyData PDX (virtual)

February 9th 2022

Max Halford

Goals of this talk

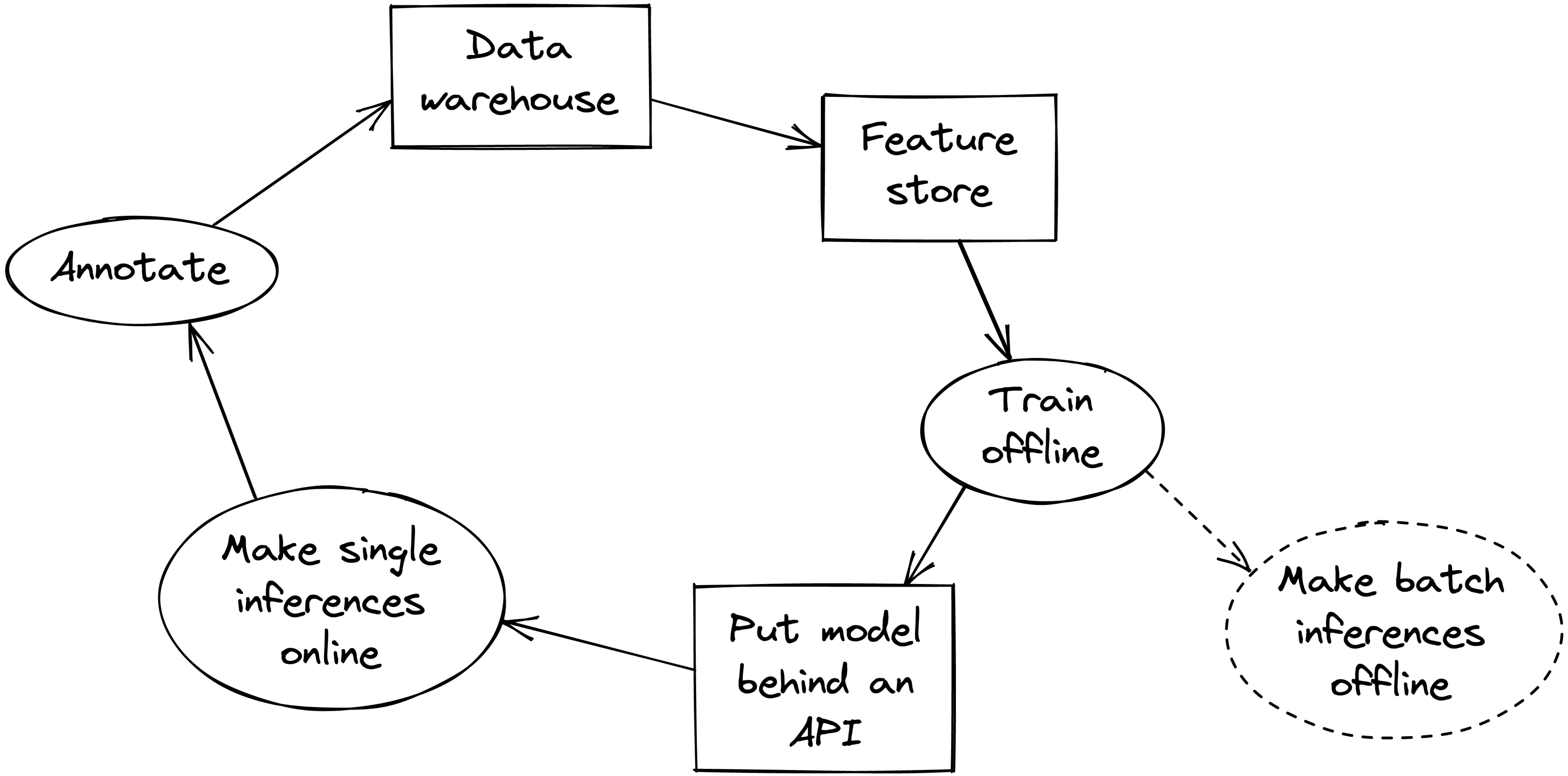
1. Get accustomed to online machine learning
2. Discuss the pros and cons
3. Present River
4. Going into production
5. The road ahead

Hello, I'm Max! 👋

- 🤓 Data scientist at Carbonfact
- 😴 Online ML is a hobby
- 🥵 Kaggle competitions master
- ✍️ I like to blog
- 🇫🇷 Mostly based in France
- 😊 Very happy to be here!

A BRIEF INTRO TO ONLINE MACHINE LEARNING

The machine learning we're used to



The limits of batch learning

A batch model has to be retrained from scratch to learn from new data, meaning:

😬 It can't adapt when new classes appear

⏱️ It takes time

🔌 It's wasteful

👯 Online/batch parity is not ensured

Case example: customer service automation

1. Customers ask your customer service questions
2. You automate that with predefined answers (templates)
3. Answers are suggested by a model based on the conversation
4. Formulate it as a multi-class classification problem
5. The product and the questions evolve, so you add templates
6. You now have to wait for the model to be retrained

"Real-time" is a weasel word

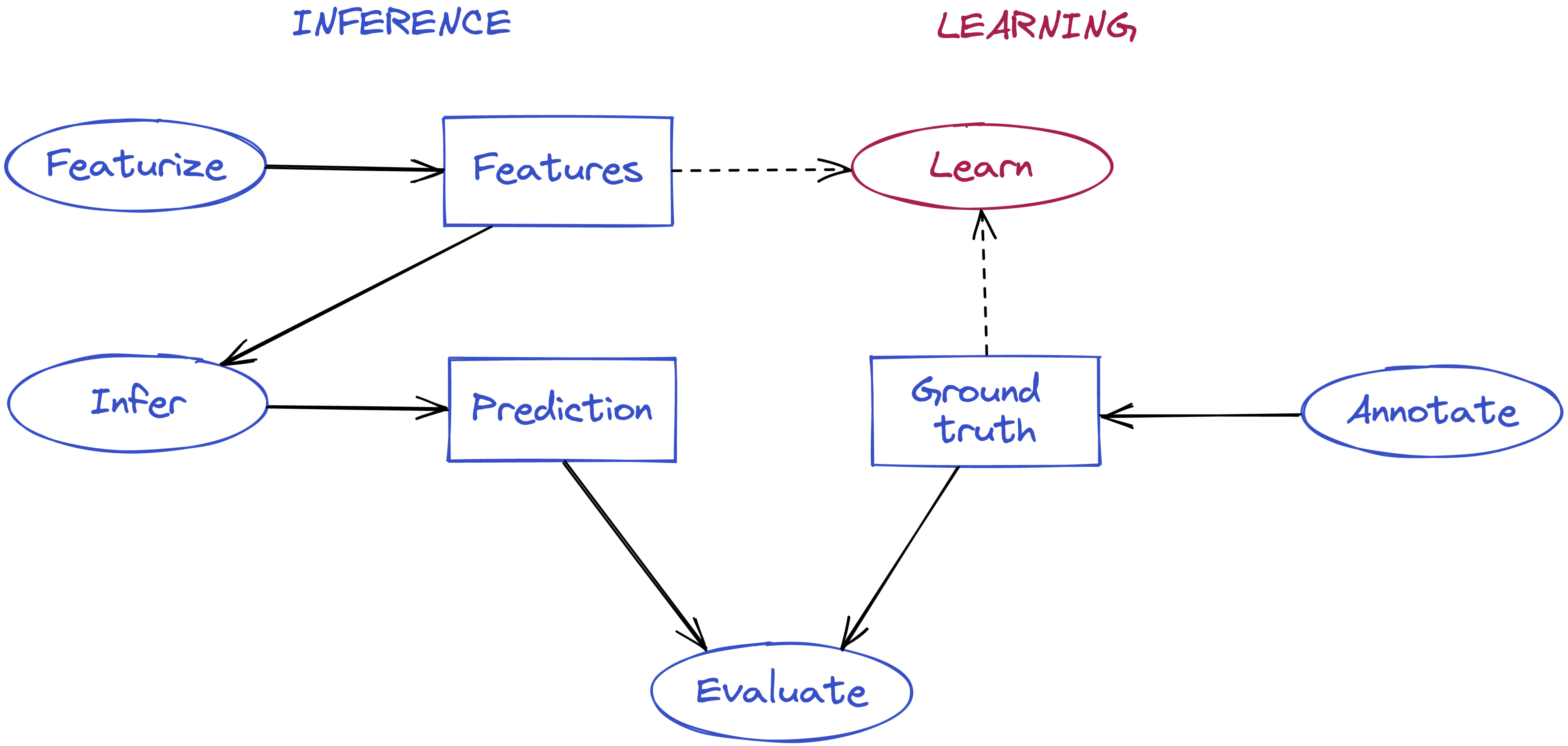
- There is no single definition
- Real-time means what you want it to mean
- Different applications will have different requirements
- Retraining a batch model periodically might be enough
- What matters is that your model has positive business impact

Online models

An online model can learn from a stream of data.
What does that imply?

- The model can learn one sample `(x, y)` at a time
- Past samples do not have to be revisited
- The number of inputs doesn't have to be specified beforehand
- Likewise for the outputs
- The model may gradually forget, and focus on the recent past

Online learning != online inference



Online / batch parity

1. How do you make sure features are available for inference?
2. Leakage is always possible, even if you use a feature store
3. In an online scenario: you predict, and then you learn
4. You learn with the same features used for inference
5. Online/batch parity is ensured 👯
6. <u>Some</u> call this the “log and wait” approach

Progressive validation

A method for evaluating online models offline.

1. Your dataset is a stream `(x, y)`
2. First, you predict the output of features x
3. Then you update the model with `(x, y)`

You can do this for the whole dataset, which means:

A. The model is fully trained
B. The validation score is calculated over the whole dataset
C. You’ve reproduced what happens in practice

Online machine learning: the pros...

- 🌱 It's ecological because past data isn't revisited
- ⚡ The model is as up-to-date as possible
- 🎯 Online/batch parity is ensured
- 🤝 Backtesting is reliable
- 🧙 It feels like magic when it's working

… and cons

🤷‍♀️ Not many people know about it

📚 It hasn't received as much research attention as batch learning

⚒️ There isn't as much tooling around it

✨ It requires a paradigm shift

River, and
why we made it

How I got started

- I got curious about online machine learning in early 2019
- It felt refreshing and compatible with real-world applications
- There wasn't a lot of literature or blog posts
- Existing libraries didn't seem friendly to me
- I decided to write my own library to learn, it was called creme

Merging with scikit-multiflow

- In early 2020 I met with the scikit-multiflow team
- They were also an online machine learning library
- They were a bit more research oriented
- We decided to merge
- It took a lot of discussion and compromise
- Our offspring is called River, and we're a happy family

River

- 🤝 The merger happened 16 months ago
- 🗺️ Core developers from 🇧🇷, 🇳🇿, and 🇫🇷
- 👷‍♀️ ~26,522 lines of code, ~2500 unit tests
- 🏭 Used in production, including at Lyft
- 🤗 Open and welcoming with contributions

Minimal example

```python
>>> from river import compose
>>> from river import linear_model
>>> from river import metrics
>>> from river import preprocessing

>>> model = compose.Pipeline(
...     preprocessing.StandardScaler(),
...     linear_model.LogisticRegression()
... )

>>> metric = metrics.Accuracy()

>>> for x, y in dataset:
...     y_pred = model.predict_one(x)      # make a prediction
...     metric = metric.update(y, y_pred)  # update the metric
...     model = model.learn_one(x, y)      # make the model learn

>>> metric
Accuracy: 89.20%
```

No numpy, no PyTorch, just dictionaries

```
>>> from pprint import pprint
>>> from river import datasets

>>> dataset = datasets.Phishing()

>>> for x, y in dataset:
...     pprint(x)
...     print(y)
...     break
{'age_of_domain': 1,
 'anchor_from_other_domain': 0.0,
 'empty_server_form_handler': 0.0,
 'https': 0.0,
 'ip_in_url': 1,
 'is_popular': 0.5,
 'long_url': 1.0,
 'popup_window': 0.0,
 'request_from_other_domain': 0.0}
True
```

It's a general purpose library

Bandits

Ranking

Preprocessing

Online statistics

Feature extraction

Anomaly detection

Multi-output learning

Random forests

Naive Bayes

Clustering

Time series forecasting

Online metrics

Linear models

Nearest neighbors

Imbalanced learning

Model selection

Decision trees

Factorization machines

Neural networks

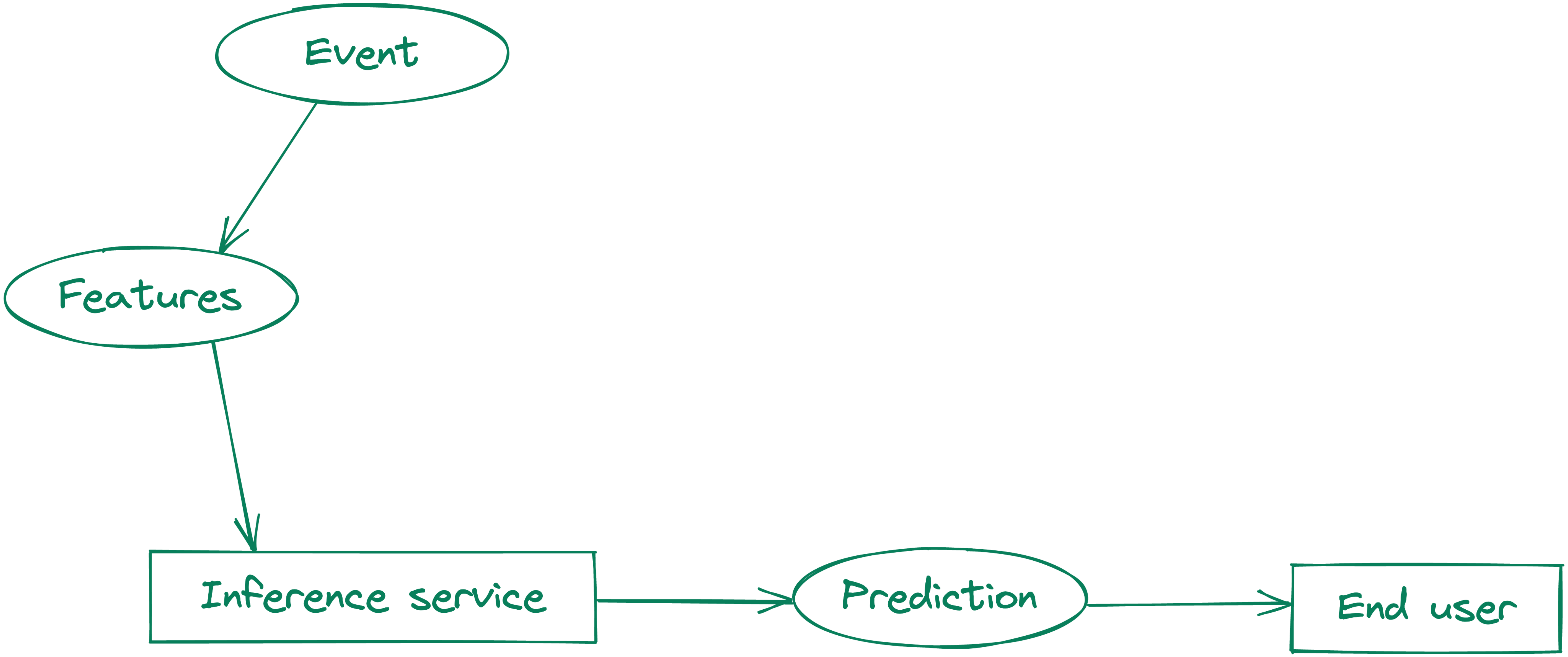
What River is not

- River is a library of online machine learning algorithms, period
- Think of it as the equivalent of sklearn for online learning
- *(though that it’s a bit of an ambitious statement)*
- River is not an MLOps tool
- We see people integrate River into their systems in many different ways
- There is the need/opportunity to build a higher level tool for online machine learning MLOps

Online machine learning MLOps

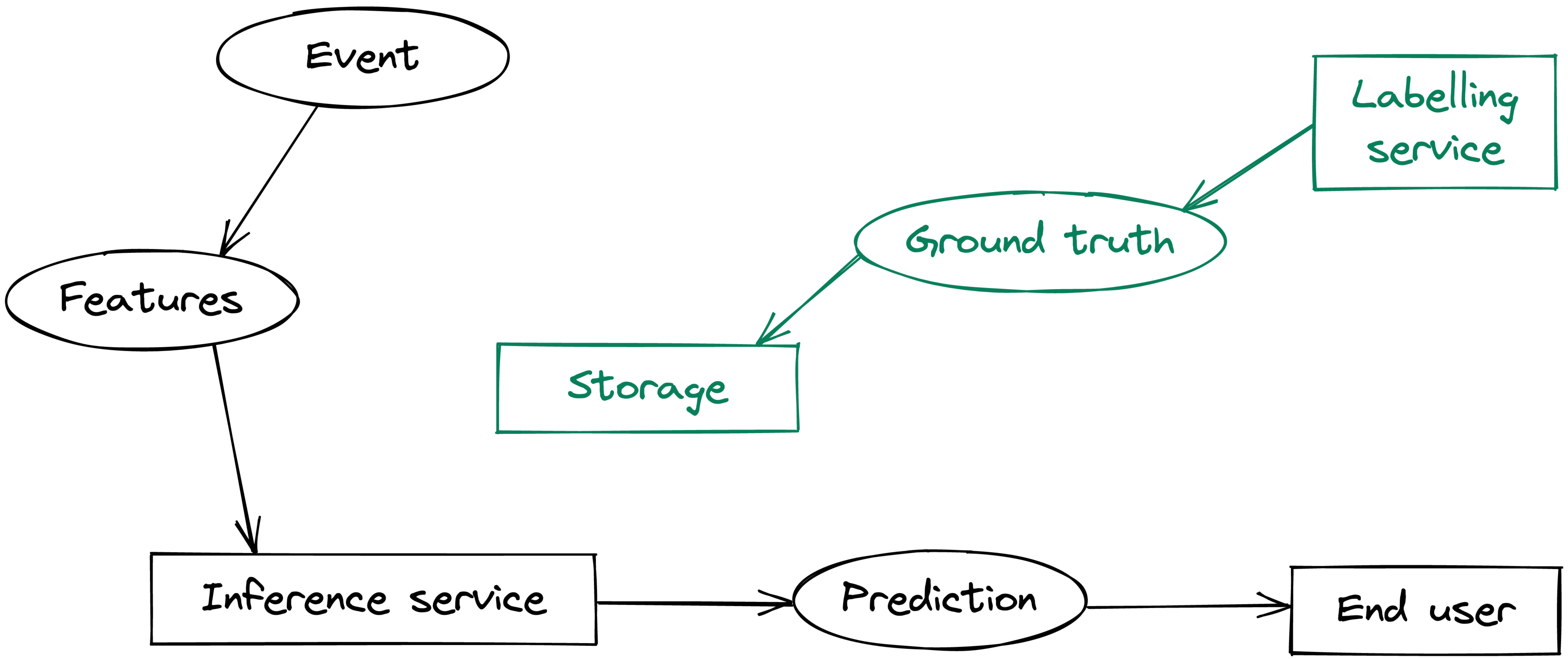
Inference service

- Let’s say we’re building an MLOps system
- The most basic requirement is to be able to make inferences
- This is no different to batch machine learning
- Online models can be scaled across multiple machines
- We distinguish events from features
- Events are turned into features that can be fed to a model


Event
Features
Inference service
Prediction
End user

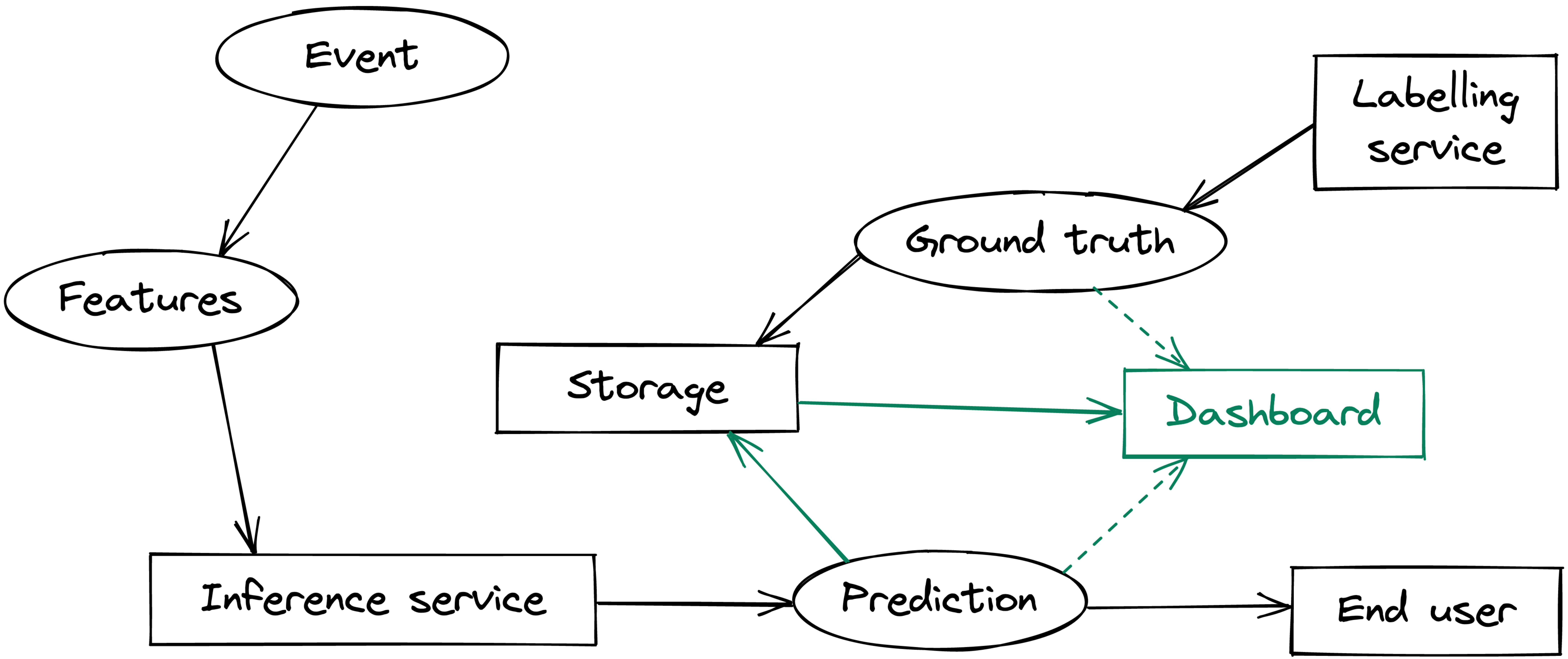
Labelling service

- We need to record labels before we can do anything else
- There are two types of labels
  A. Natural labels produced by the environment
  B. Labels that require human intervention
- This isn't an MLOps task per say
- But an MLOps system needs to be able to receive labels


Event
Features
Labelling service
Ground truth
Storage
Inference service
Prediction
End user

Performance monitoring

- We have predictions and labels
- The first thing we can do is measure the model's performance
- Ideally, we want to build a real-time dashboard
- We also want all this data to be available programmatically

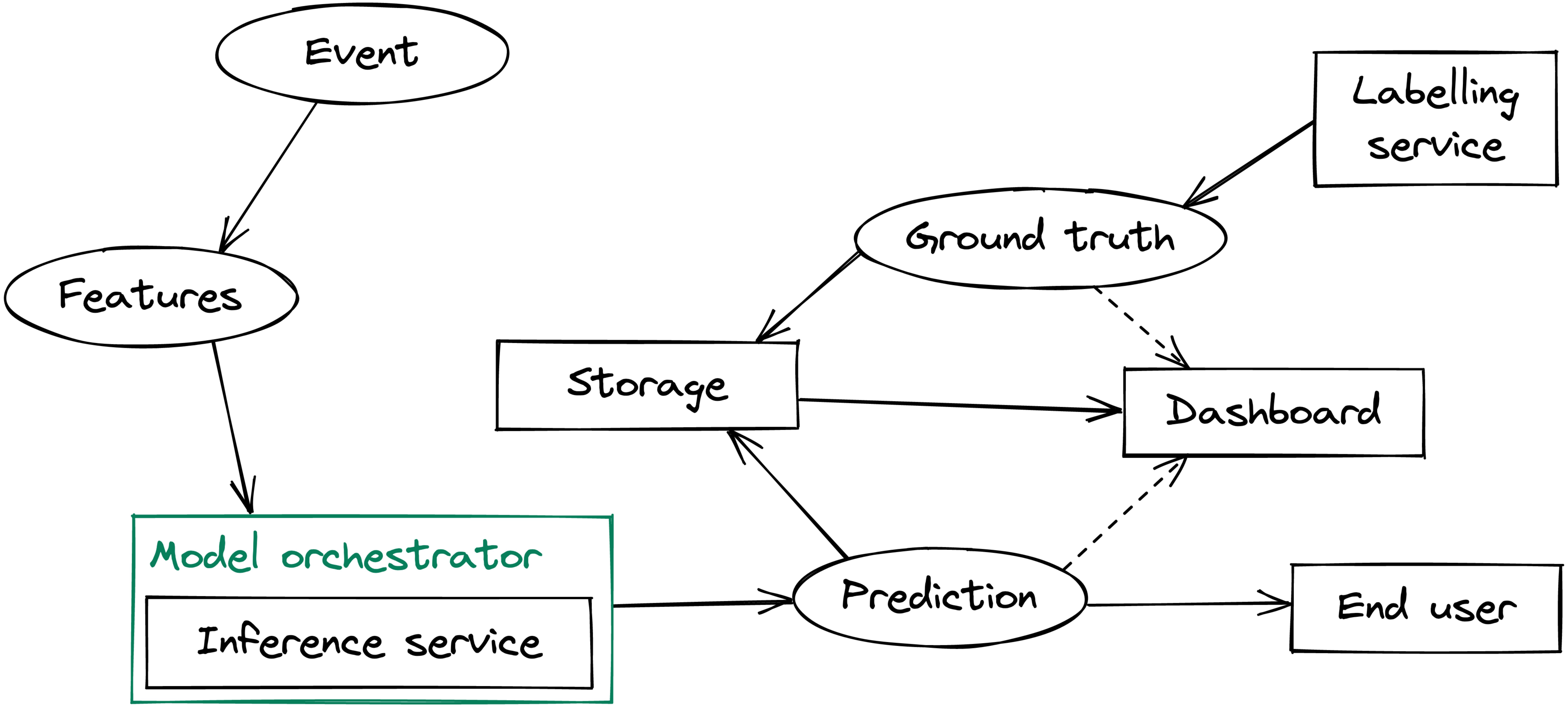

Event
Features
Inference service
Prediction
End user
Storage
Ground truth
Labelling service
Dashboard

Model orchestration

- In reality, there is more than one model
- You might want to change the model's architecture
- You might want to add or remove a feature
- You can test offline which configuration is the best
- But ideally you should be able to A/B test online
- For this you need a model orchestration mechanism

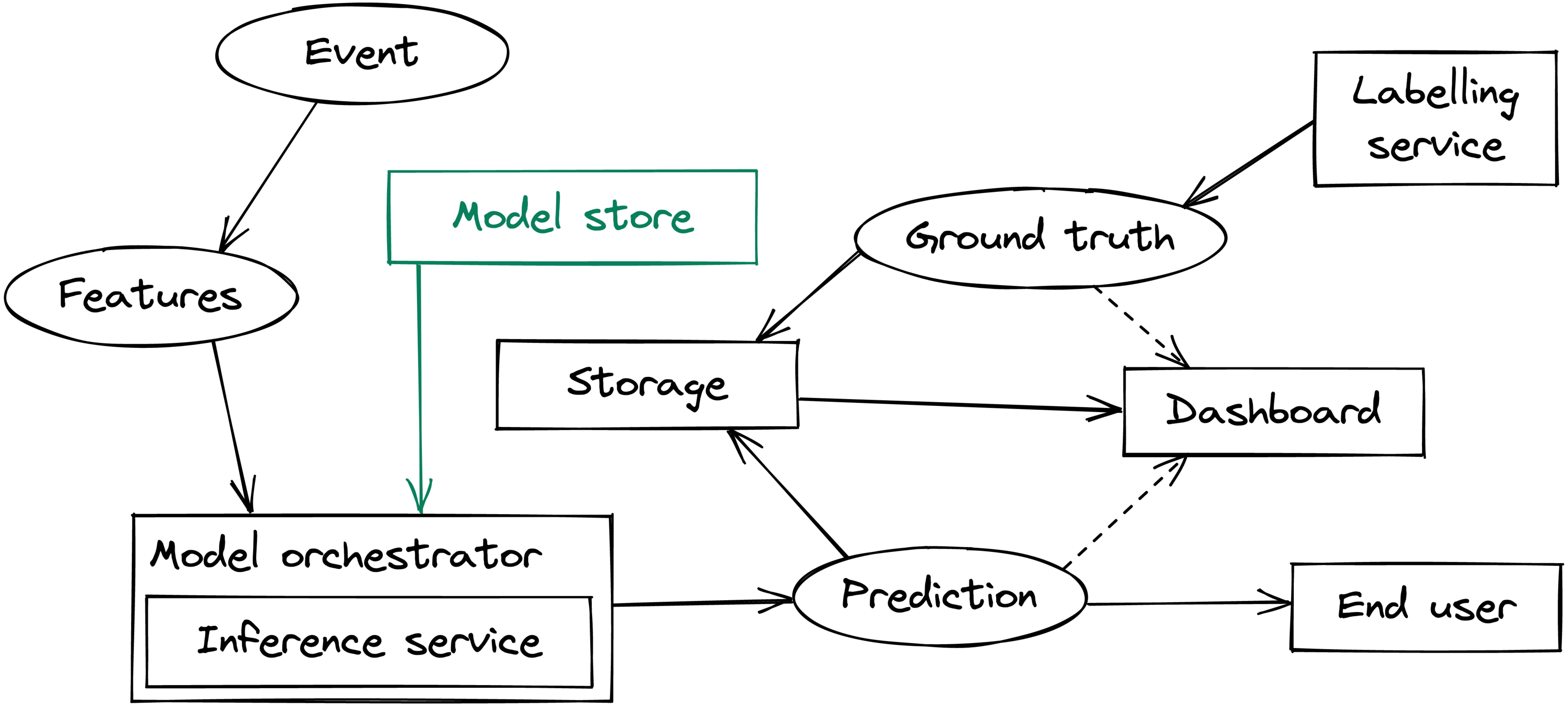
Shadow deployments 👻

- This is the simplest orchestration mechanism
- You simply ask each model to make a prediction
- You compare each prediction with the ground truth
- The best model’s predictions are the ones that are served
- The other models are said to be “shadowing”
- In other words: you don’t need A/B tests or canary deployments


Event
Features
Model orchestrator
Inference service
Prediction
Storage
Ground truth
Labelling service
Dashboard
End user

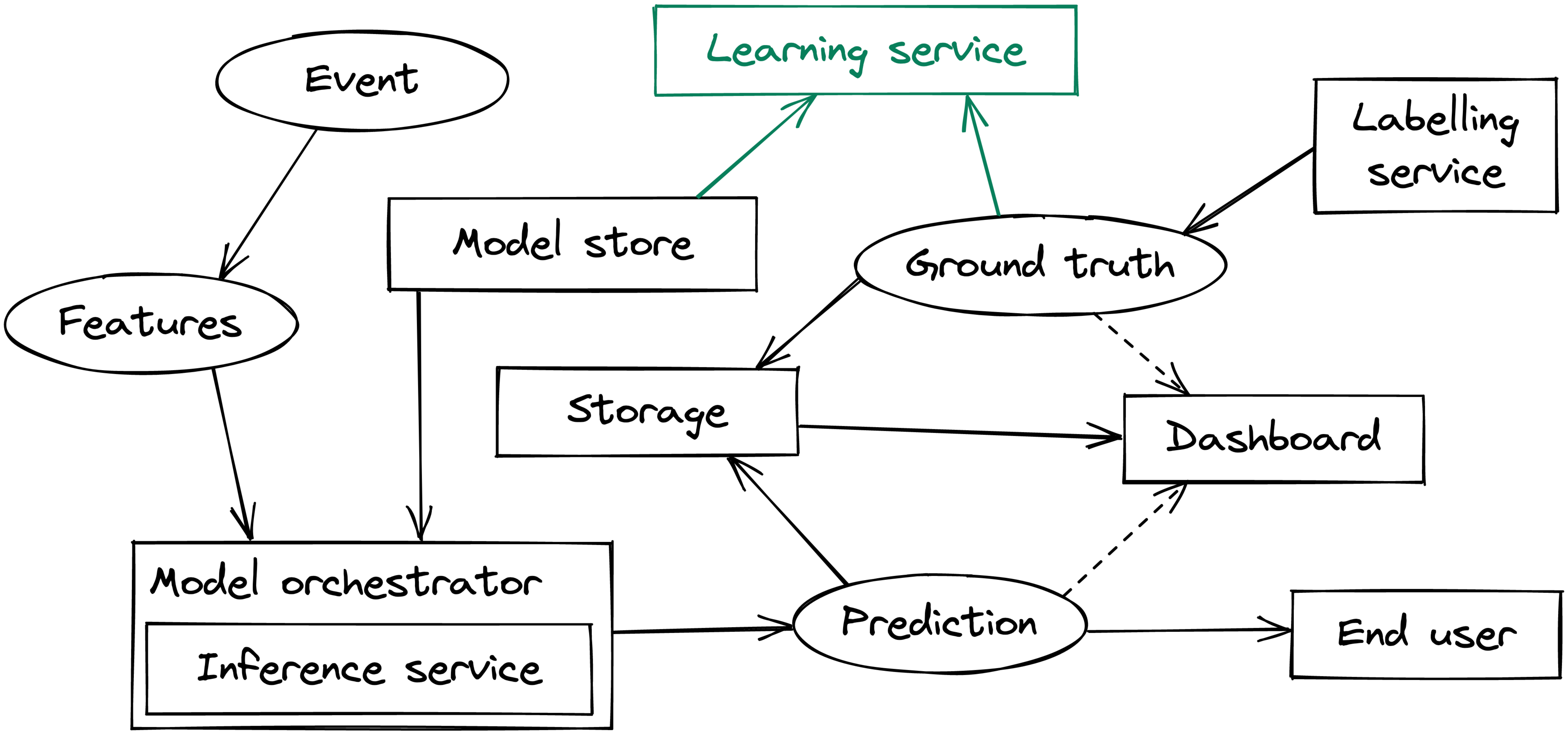
Model storage

- You're going to need a tool to manage your models
- A model store is essentially a database for models
- The inference service will query this model store


Event
Features
Model store
Model orchestrator
Inference service
Storage
Ground truth
Labelling service
Dashboard
Prediction
End user

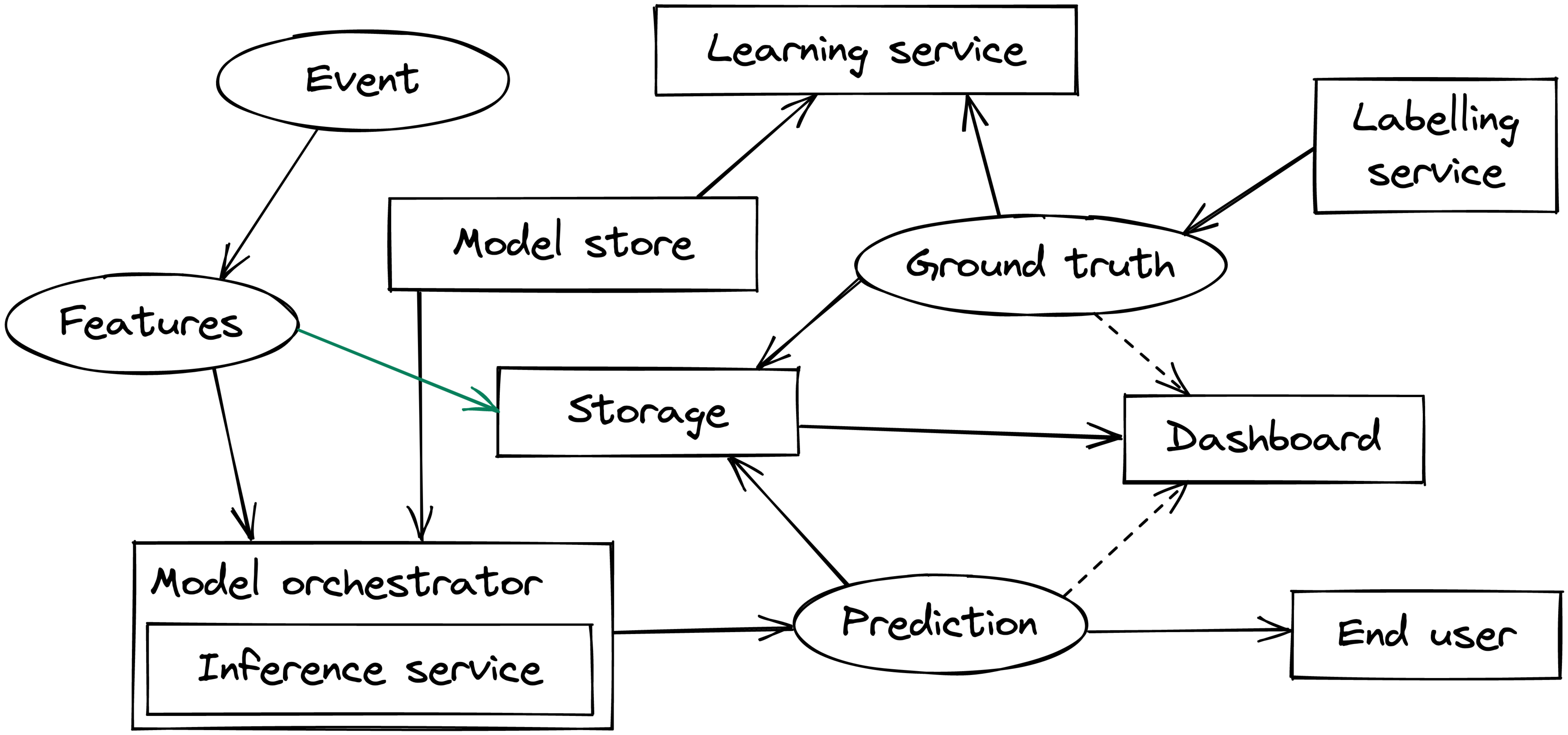
Learning service

- Now we’re ready to train our models
- We already have a model store
- We also store the ground truths
- Training can be done each time a ground truth is revealed
- You can also do it periodically
- In any case, learning is a bottleneck that can’t be distributed


Learning service
Event
Labelling service
Model store
Ground truth
Features
Storage
Dashboard
Model orchestrator
Inference service
Prediction
End user

"Log and wait" storage mechanism

- Ideally we should use the features we used for inference
- This requires storing the features in a database
- The features are joined with the ground truth when it arrives
- This implies an identification system


Event
Learning service
Labelling service
Model store
Ground truth
Features
Storage
Dashboard
Model orchestrator
Inference service
Prediction
End user

Next steps

We didn't talk about technologies

- The system we described is a blueprint
- Different pieces of technology may be used for each part
- Ideally, users should be able to use their own infra
- For instance, data teams may already have a model store

There's a lot to do

- I wrote a prototype called Chantilly
- It's a bit (too) simple, it's just a Flask app
- There's a lot to do to get something people can use like River
- But we do see a need from community, so that's motivating
- Goal: find some time in 2022 to do some deep work on this

Real-world use cases are necessary

- This system is more holistic than River
- It's difficult to satisfy everyone's needs
- Solving 80% of use cases would be nice
- It needs to be battle-tested in production
- I'm very much open to collaboration 🤗

Thank you

for having me!