The challenges of online machine learning in production

Or how to rethink MLOps

Max Halford — Itaú Unibanco — 2020.02.26

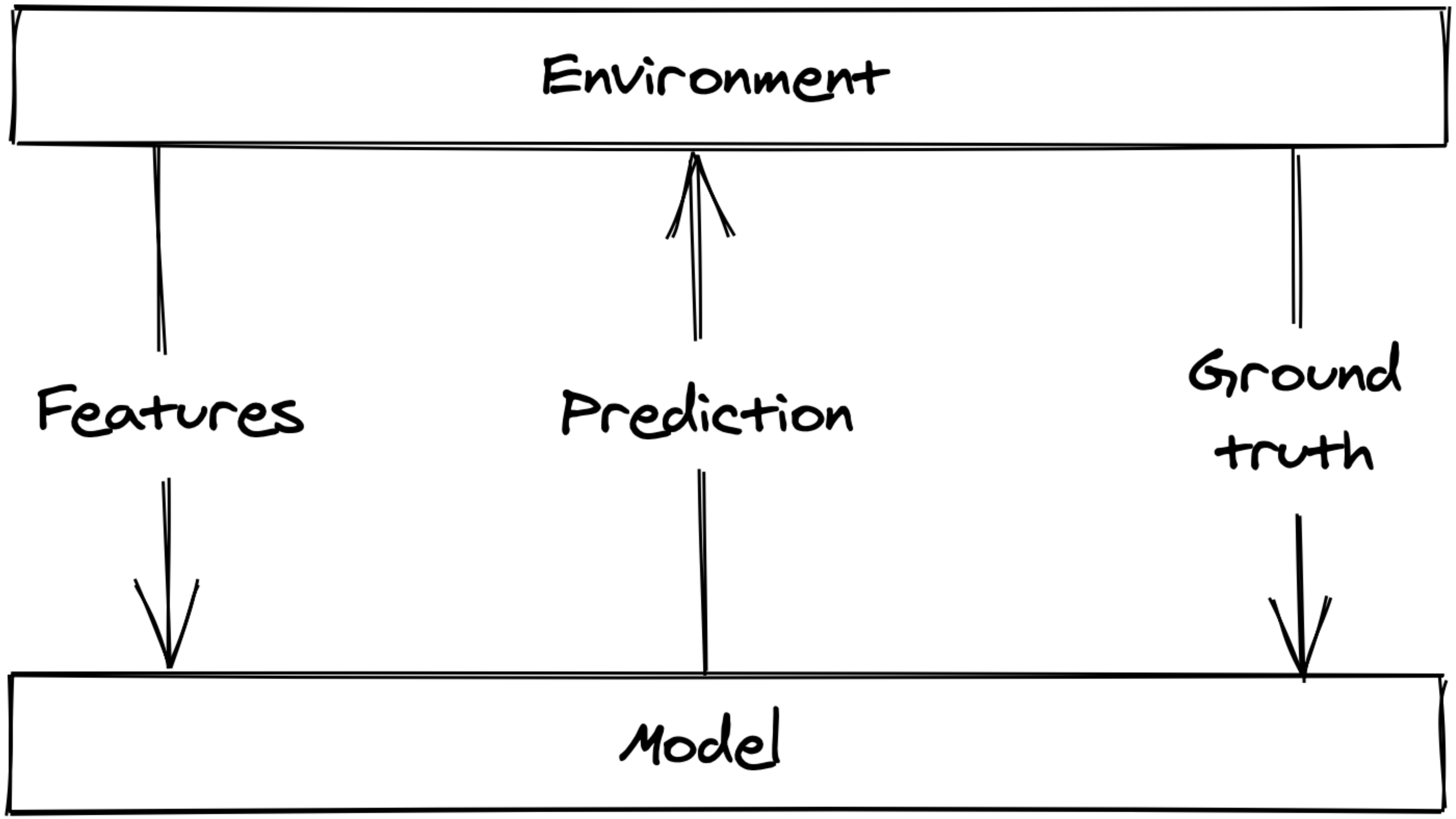
Hello 👋

- I work at Alan
- PhD in query optimisation
- Kaggle competitions Master
- I ❤️ open source software
- I like to blog

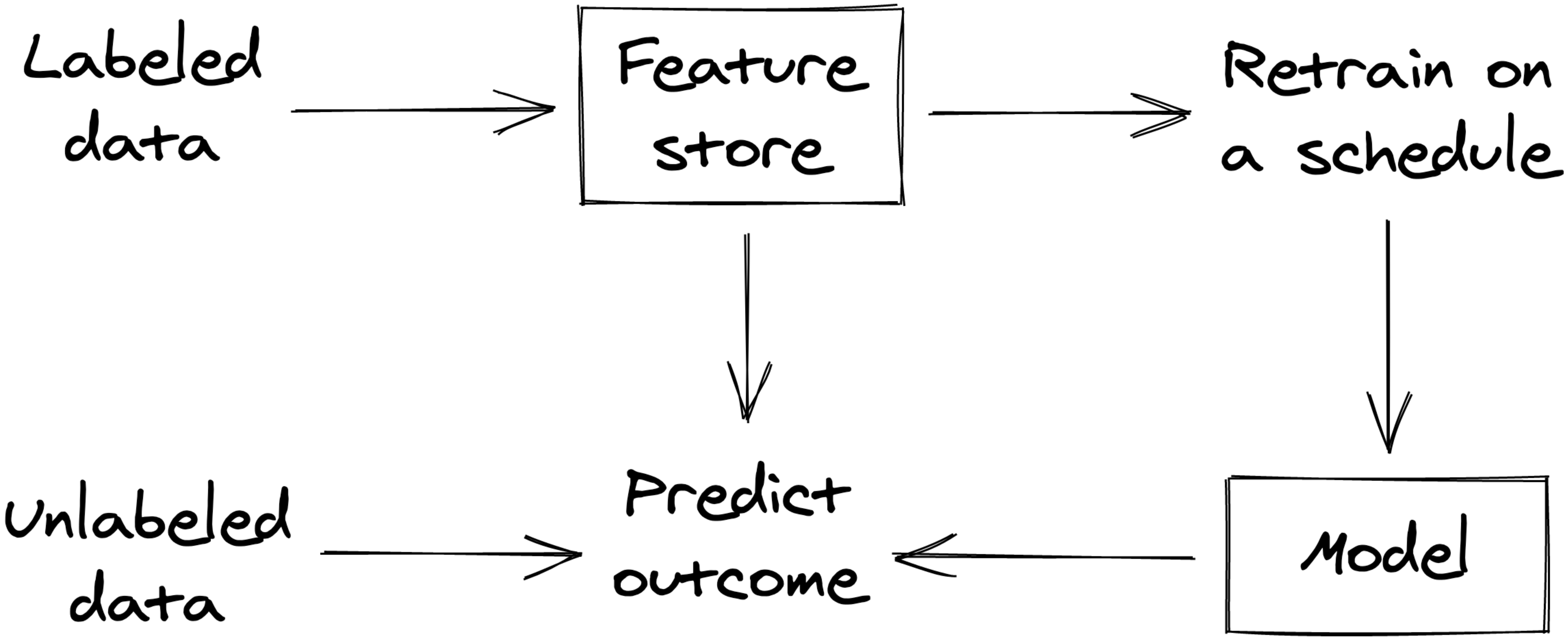
Machine learning

- (Supervised) ML is about:
    1. Finding patterns in labeled data
    2. Predicting the outcome of unlabelled data
- Most ML models are batch models
- Batch models are trained on a static dataset

The real world is dynamic



MLOps as we know it

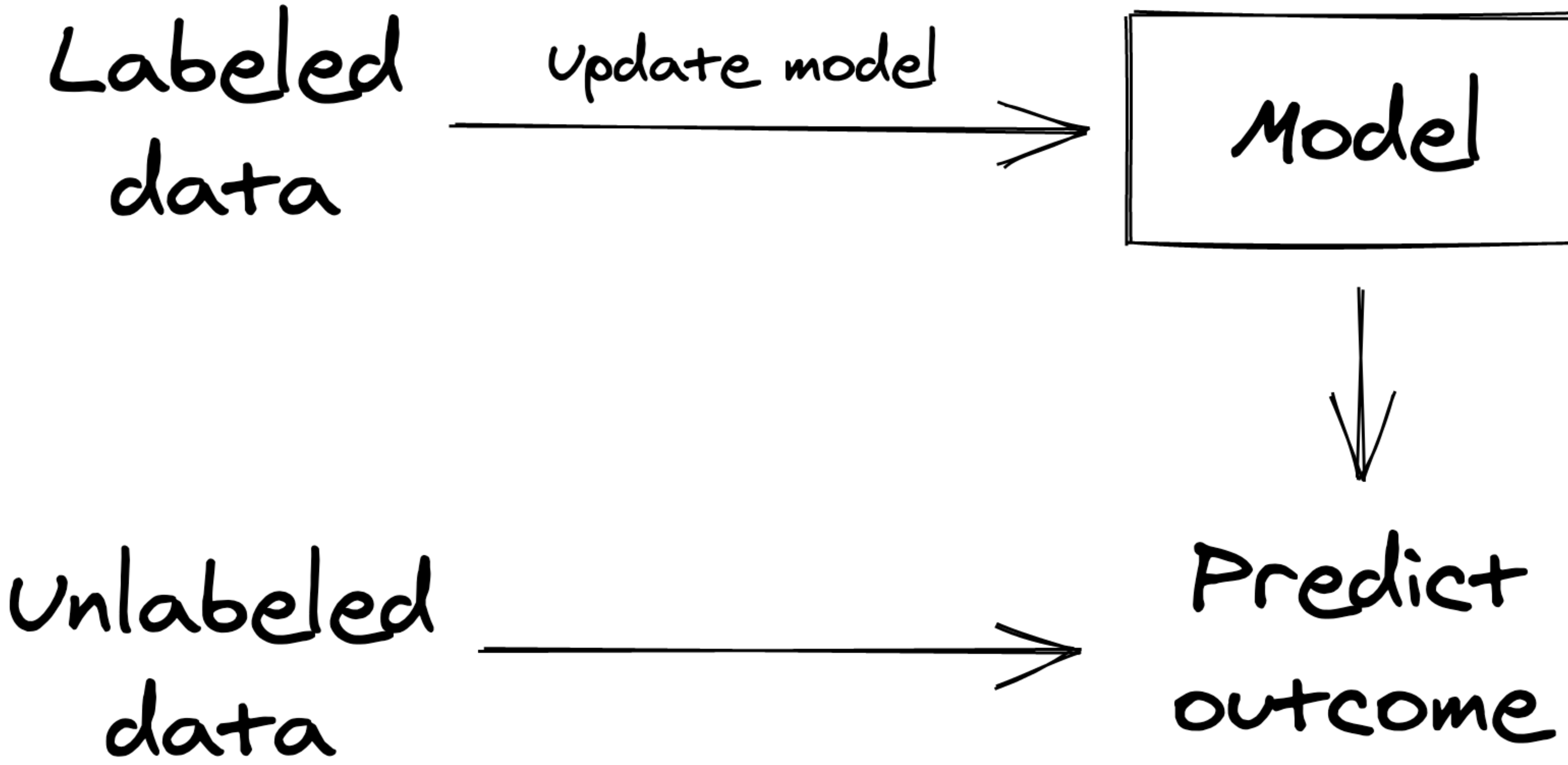


(this is called a Lambda architecture)

MLOps challenges

- *How often should I retrain my model?*
- *Am I backtesting my model correctly?*
- *What if I want to update my model ASAP?*
- *Am I sure my features can be used at prediction time?*

MLOps with an online model



(this is called a Kappa architecture)

The benefits

- The model is (almost) always up-to-date
- Learning a new sample is cheap
- Feature extraction is reliable
- Backtesting is natural and trustworthy

The difficulties

A. Less models to choose from
B. Updating the model is a computational bottleneck
C. Smaller research field and talent pool

My mission is to provide remedies to these difficulties!

River

- Python library for online machine learning
- Merger between creme and scikit-multiflow
- Started in January 2019
- 3 core developers from 🇧🇷 🇫🇷 🇳🇿
- In use at a couple of companies

Naive Bayes

Metrics

Datasets

Kernel approximation

Rating systems

Anomaly detection

Multioutput models

Drift detection

Preprocessing

Time series

Nearest neighbours

River

Feature extraction

Boosting

Trees

Factorisation machines

Pipelines

Neural networks

Feature selection

Linear models

Bandits

Bagging

Expert Learning

Imbalanced learning

Statistical measures

A little teaser

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

X_y = datasets.Phishing()  # this doesn't put anything in memory

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

metric = metrics.ROCAUC()

for x, y in X_y:
    y_pred = model.predict_proba_one(x)  # make a prediction
    metric = metric.update(y, y_pred)    # update the metric
    model = model.learn_one(x, y)        # make the model learn
```

Do what you want!

```python
@app.route('/predict', methods=['GET'])
def predict():
    payload = flask.request.json
    x = payload['features']
    model = db.load_model()
    y_pred = model.predict_proba_one(x)
    return y_pred, 200


@app.route('/learn', methods=['POST'])
def learn():
    payload = flask.request.json
    x = payload['features']
    y = payload['target']
    model = db.load_model()
    model.learn_one(x, y)
    return {}, 201
```

Motivation

- River is great and users seem to **enjoy it**
- But there's no obvious way to deploy online models
- Online training involves a unique set of challenges
- We want to **delight** users

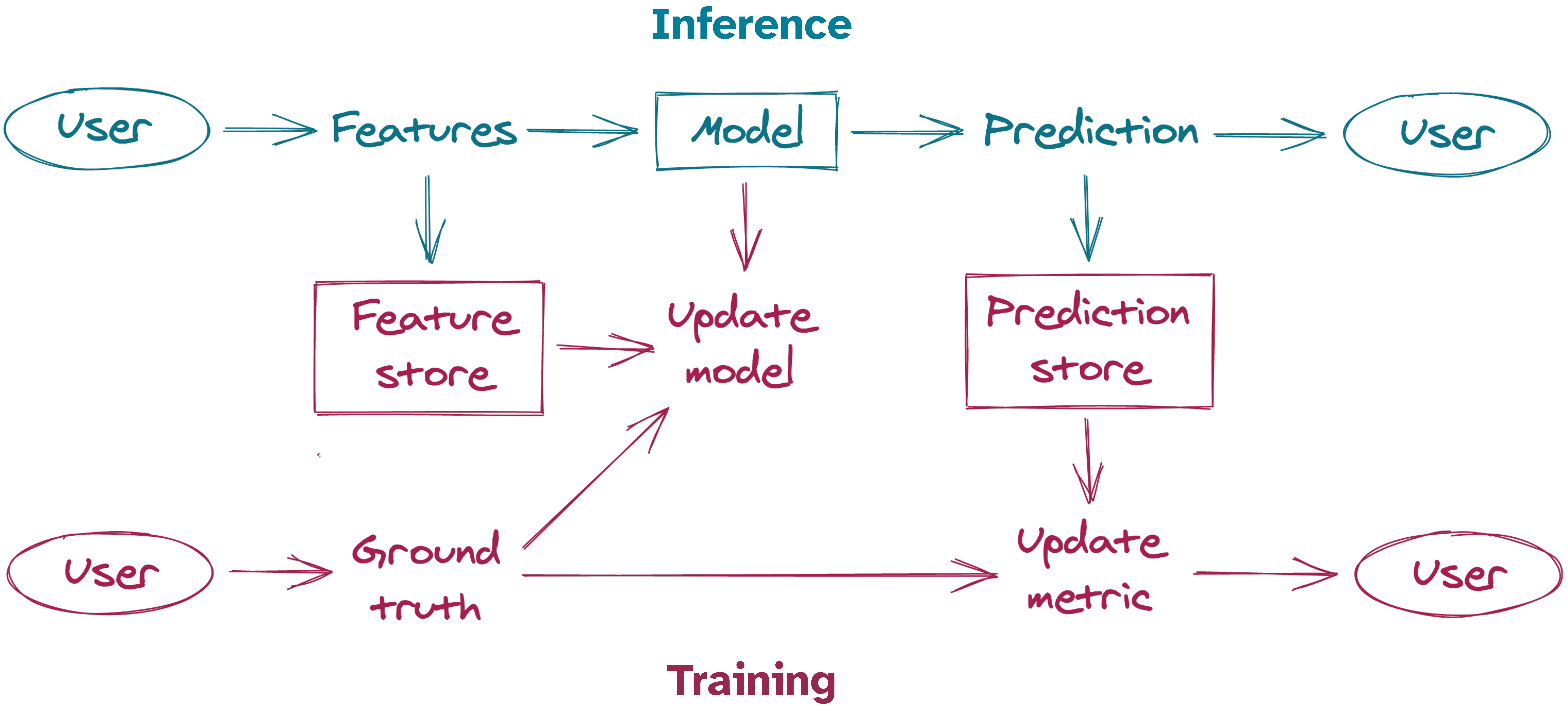
Scaling predictions is easy...

- ... if models are static
- You just have to provision multiple workers
- Each worker is provided with a copy of the model

Auto-scaling is a convenience in 2021!

cortex Kubeflow dataiku H2O.ai BENTOML

Online learning process



Online model training

- Features **x** are stored aside when a prediction is made
- Ground truth **y** is attached to features on arrival
- A stream of **(x, y)** pairs is fed to the model
- Essentially, this is an infinite while loop

If there's too much velocity, what happens?

It depends on the queuing system

- First in, first out (FIFO)
  - The model trains on old data first
  - The model will miss out for a while on new data
- Last in, last out (LIFO)
  - The model trains on the latest data first
  - The model might never see some data

Speed considerations

- Many libraries implement SGD, allowing for comparison
- River shines in single instance processing:
  - 5x faster than Vowpal Wabbit
  - 20x faster than scikit-learn
  - 50x faster than PyTorch
  - 180x faster than Tensorflow

Mini-batching

```
model = (
    preprocessing.StandardScaler() |
    neural_network.MLPRegressor(
        hidden_dims=(10,),
        activations=(
            nn.activations.ReLU,
            nn.activations.ReLU,
            nn.activations.ReLU
        ),
        optimizer=optim.SGD(1e-4),
        seed=42
    )
)

dataset = datasets.TrumpApproval()
batch_size = 32
for epoch in range(10):
    for xb in pd.read_csv(dataset.path, chunksize=batch_size):
        yb = xb.pop('five_thirty_eight')
        y_pred = model.predict_many(xb)
        model = model.learn_many(xb, yb)
```

Progressive validation

- Cross-validation is ubiquitous in batch ML
- With online ML, we have a more adequate tool
- Each sample is used for prediction and then training
- The model is validated on all the data!
- Delaying ground truth arrival allows re-enacting the past

maxhalford.github.io/blog/online-learning-evaluation

github.com/online-ml/chantilly/tree/master/examples/taxis

What if I want to deploy a new model?

- Say you have a model A, and implement a new model B
- Both A and B can make predictions and get trained
- Initially, only A's predictions are sent to the user
- If B outperforms A, B's predictions can be sent instead
- This can be generalised: **bandits** and **expert learning**
- Akin to canary deployment

Some notes on existing tools

RabbitMQ

Flink

amazon Kinesis

kafka

Materialize

zeek

redis

ksqlDB

Next steps

- This architecture looks good on paper
- It needs implementing
- Technologies have to be chosen
- We don’t want to reinvent the wheel
- We want to embrace the existing ecosystem
- We’re talking to actors in the field and companies

Thank you

maxhalford.github.io

github.com/MaxHalford

maxhalford25@gmail.com