

Clojure/Conj 2024

Deploying ML models in a Clojure
environment

Oct 2024

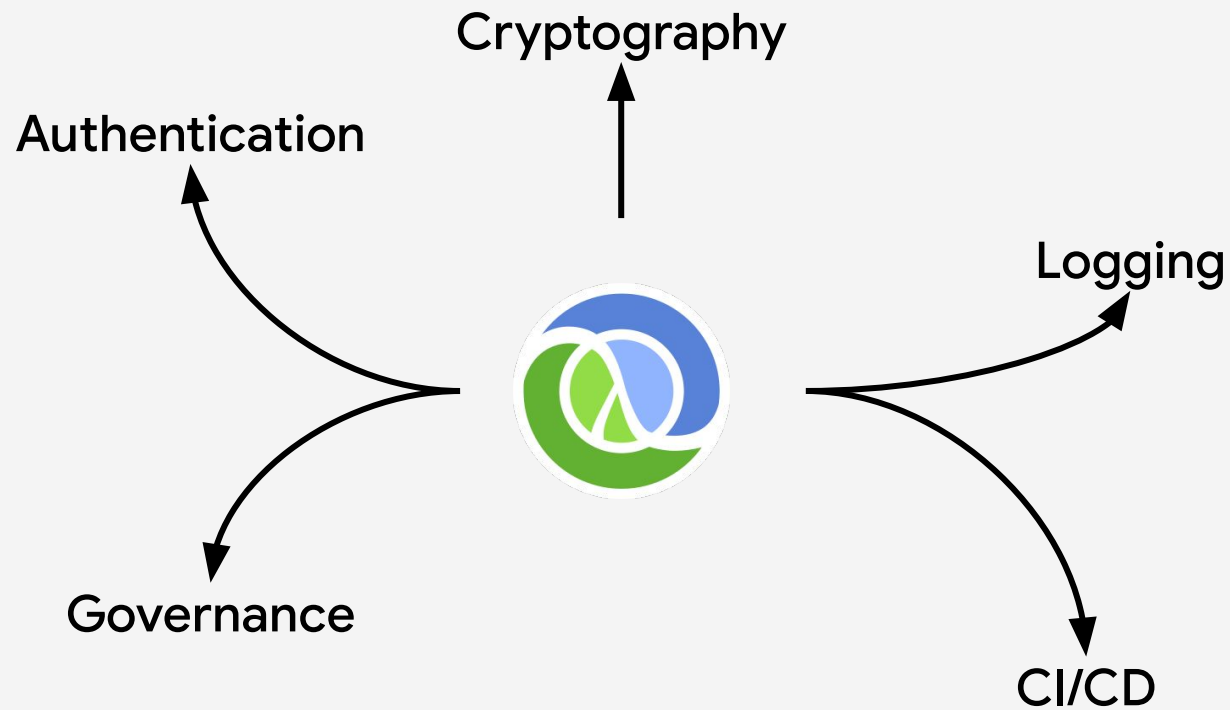


Context

We value a lot canonical approaches

Let's take a closer look on what we define as a microservice in our infrastructure and the key responsibilities it handles.





What do we do when the software is written in a different programming language?

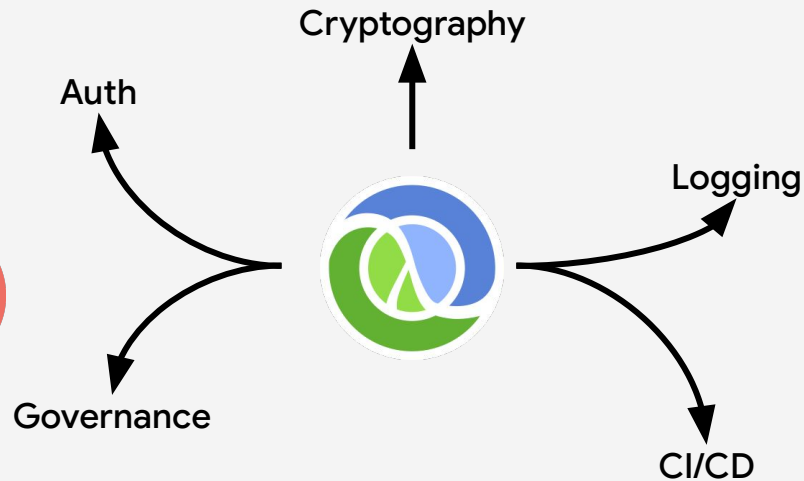
Some of the pieces stop to fit together

Typically, **auxiliary libraries** for tasks like authentication, encryption, and logging are written in Clojure. Initially, **we had to rewrite** some of this logic in other languages to ensure seamless integration with our infrastructure.

What happens in the AI/ML scope?



Machine learning



Common Clojure
microservice

Problem

AI/ML
environment
don't have the
same level of
support

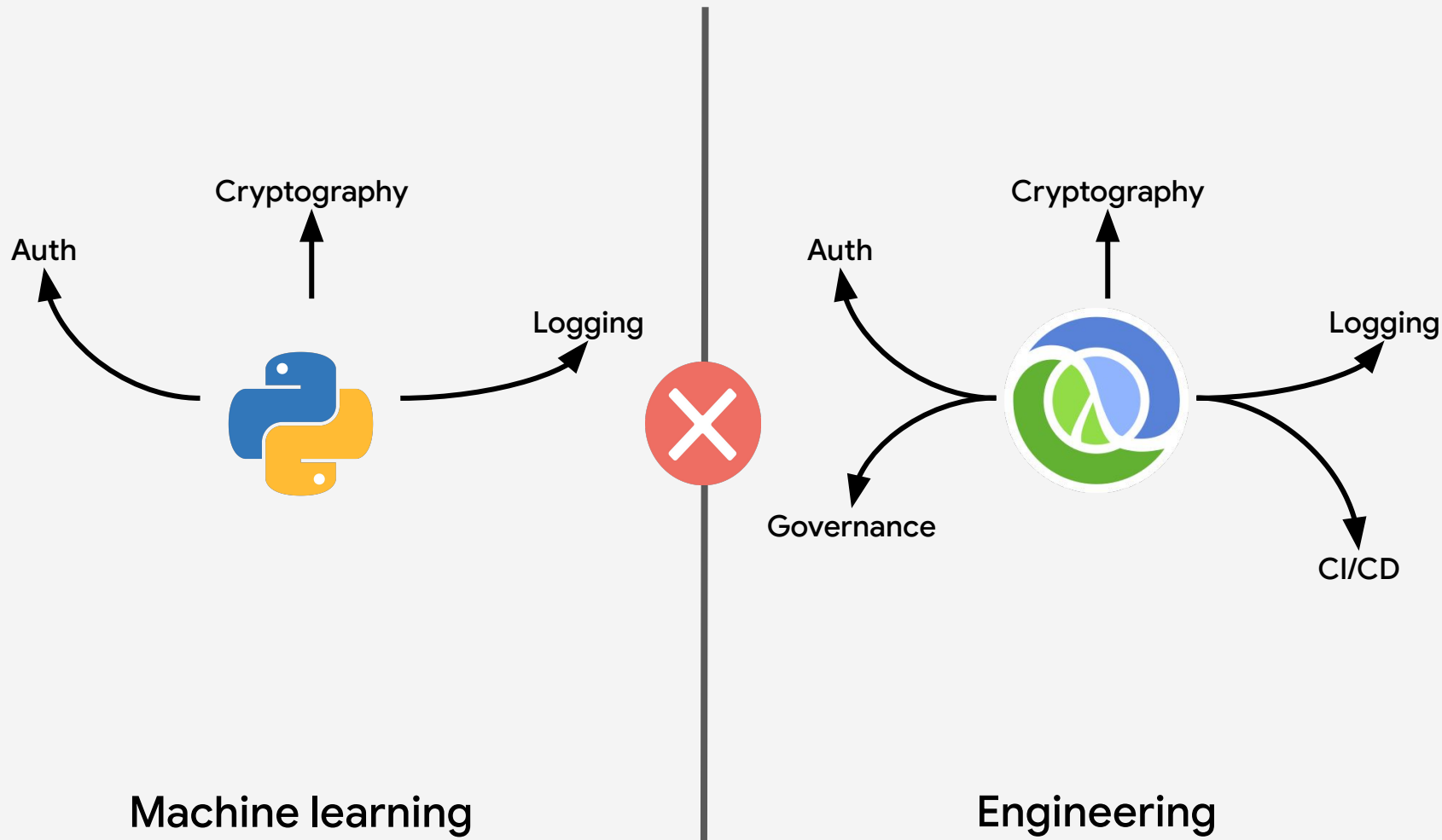
When we try to deploy a model using Python, R, Julia we find an environment that **lacks some key functionalities** available in the **canonical approach**.

Some approaches we considered:

- Rebuild libraries in python
- Use Clojure to to train/serve models

How did we solved it?

First approach



How did we solved?

First approach

Since the industry adopted Python as main programming language for AI/ML we end up recreating some of the libraries we had for Clojure using Python.

Costs

Mainly engineering hours were invested in rebuilding existing stuff.

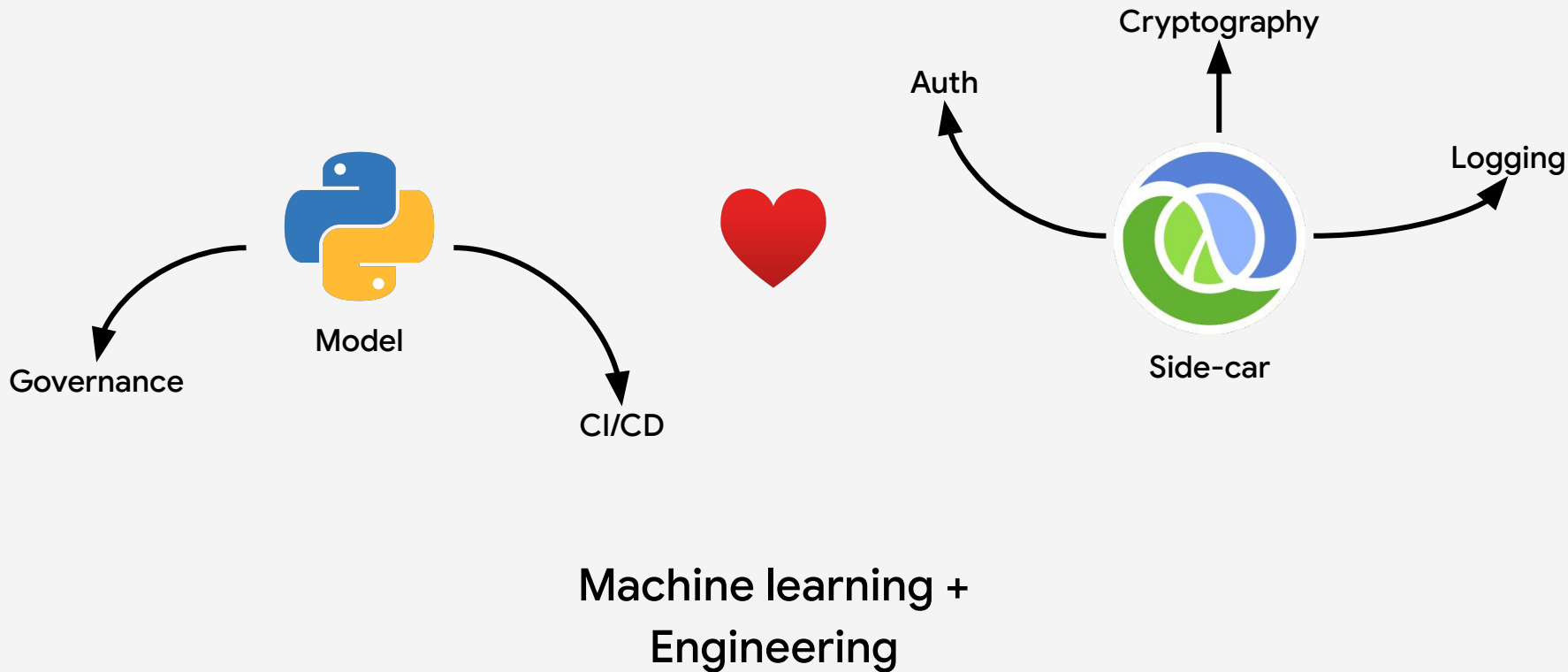
Incomplete support

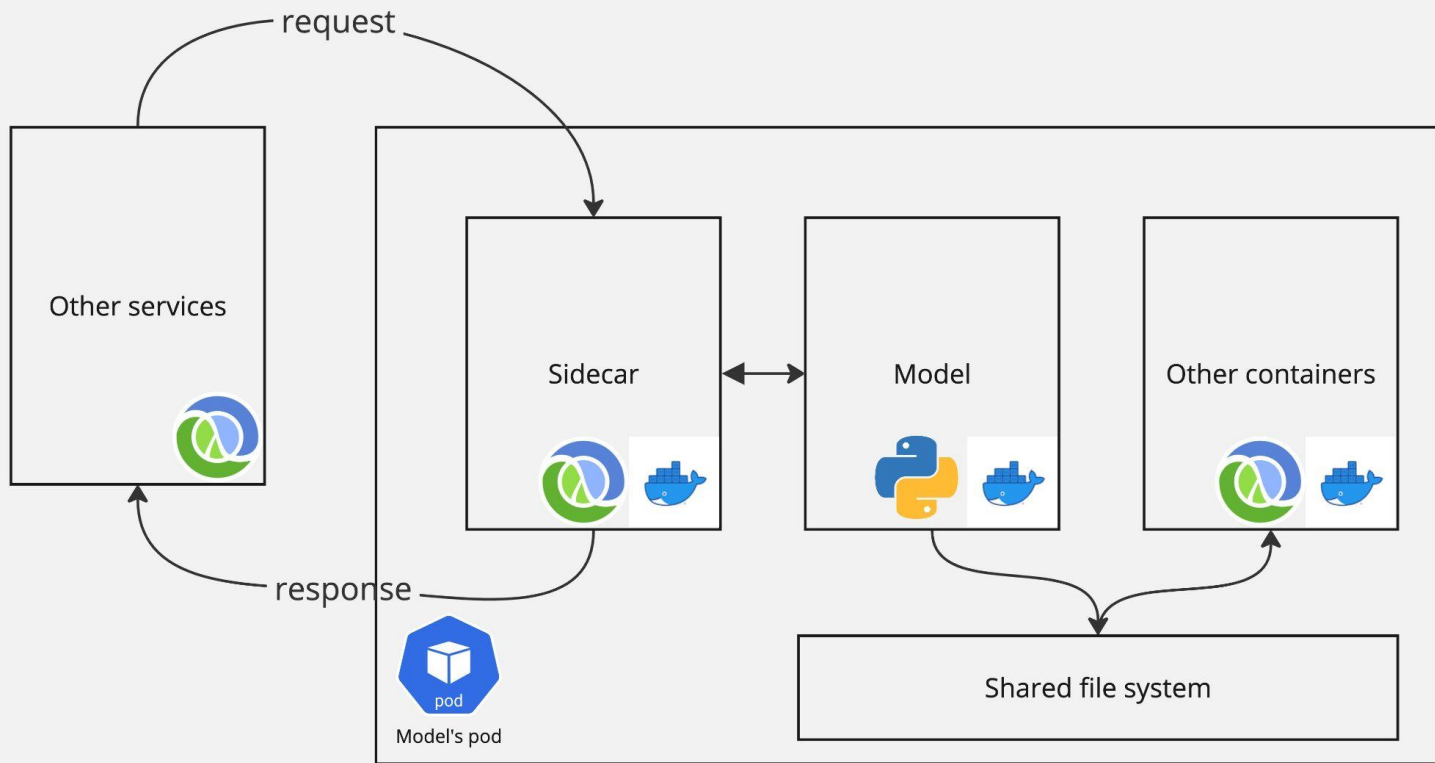
It was hard to maintain both environments with the same capabilities.

Breaking changes

The team that supported the core Clojure libraries wasn't in sync with the Python libraries, so we often faced incompatibilities.

Current approach





**Machine learning +
Engineering**

How did we solved?

Current approach

Today we rely on a Clojure sidecar responsible for being the interface between the Python model and our other microservices.

Reduce duplicated capabilities

We removed code that was used to replicate some of the things that happened in Clojure services.

Simplifies AI/ML model code

Model inference code was simplified since we don't need to deal with the complexity of communicating outside of localhost.

Integration tests

We still struggle with integration tests, leading mostly to data format-related errors in prod.

But...do we have other ways to
solve it?

Yes!!! Let's talk about
interoperability

What is interop?

What is interop?

How to make
different
systems to
communicate
between
themselves?

Interoperability is a characteristic of systems to **communicate** with one another..

Usually this is used for **communication protocols** or **data formats**, as examples we can think about XML, Json, SQL ANSI.

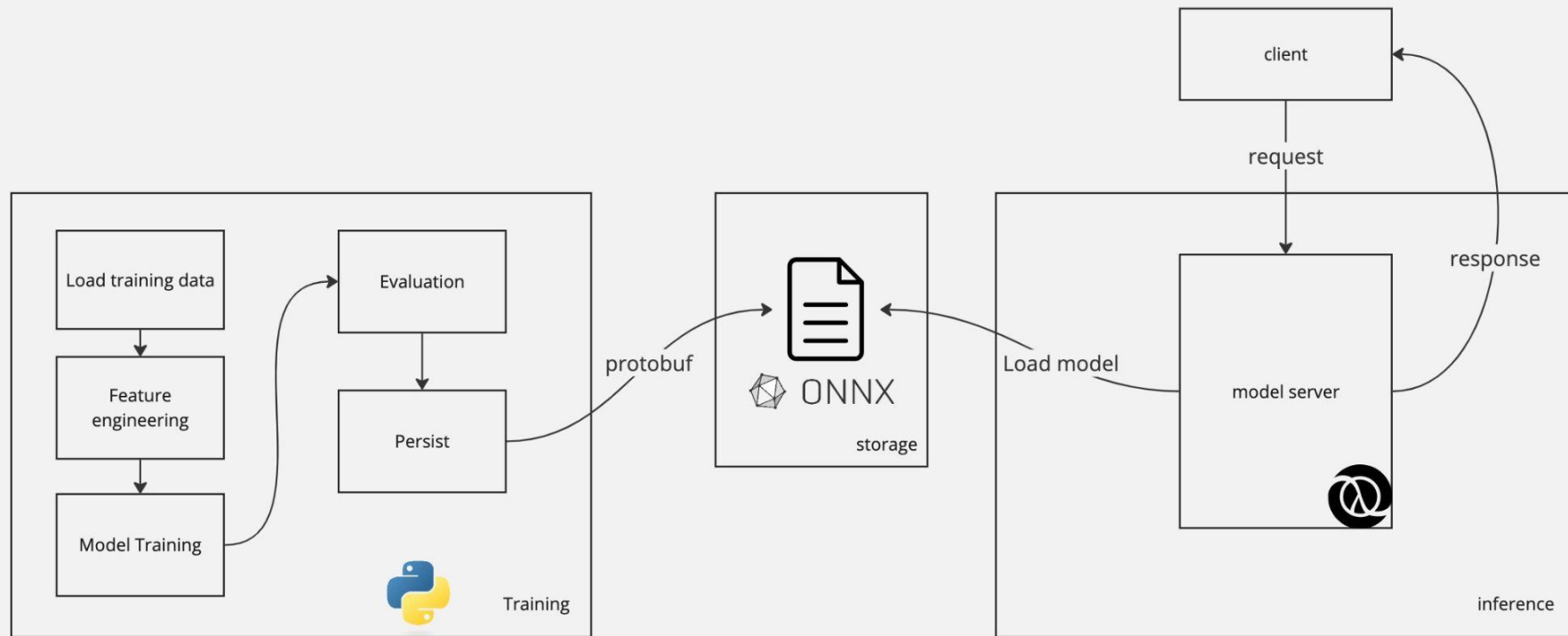
There is something like this for
AI/ML models?

What is ONNX?

What is ONNX?

Open format for AI/ML models

Open Neural Network Exchange (ONNX) is an **open format** for AI/ML models, primarily focused on the **inference environment**. This format is extensible, and a model can be represented as a directed acyclic graph (**DAG**).



What is ONNX?

Problems we can solve with it

Compatibility

Dealing with Python dependencies is a hard task. By serializing things to ONNX, we reduce our dependency to only the ONNXRUNTIME library.

Canonicity

With ONNXRUNTIME, we can design a canonical way to serve models (i.e., a Clojure component that knows how to load ONNX and run the inference).

Time-to-market (TTM)

Without the need to keep rebuilding libraries and designing different ways to run inference processes, we can reduce the models' time to market (TTM).

How to use?

How to use?

What is needed to use ONNX?

- Build models using compatible frameworks (PyTorch, TensorFlow, scikit-learn, and others).
- Serialize the model to ONNX
- Use ONNXRUNTIME to load the model
- Create either a REST API or another process to load the model and run the inference.

How to use?

Training

```
import numpy as np

from sklearn.linear_model import LinearRegression

logging.basicConfig(level=logging.INFO)

def split_data(data: np.ndarray):
    """Split data into training and testing sets."""
    from sklearn.model_selection import train_test_split
    X, y = data.data, data.target
    X = X.astype(np.float32)
    return train_test_split(X, y)

def model_train(X_train: np.ndarray, y_train: np.ndarray):
    """Train a model."""
    logging.info("Training model")
    clr = LinearRegression()
    clr.fit(X_train, y_train)
    logging.info("Model trained")
    return clr
```

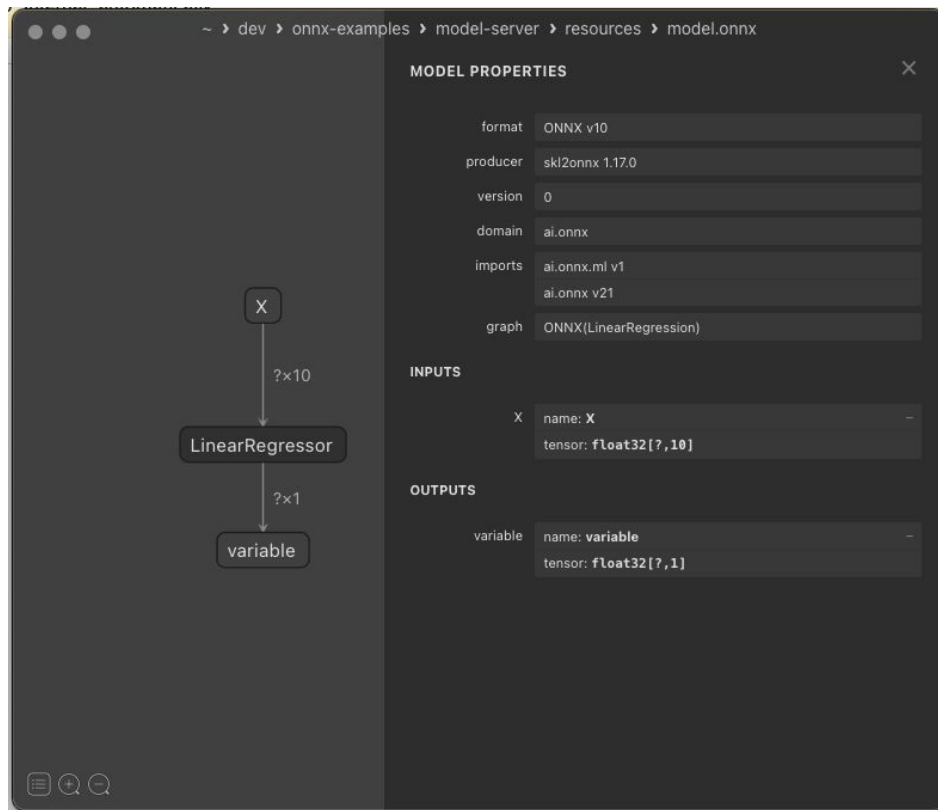
How to use?

Serialization

```
def serialize_model_onnx(clr: LinearRegression, X: np.ndarray):  
    """Serialize the model."""  
    from skl2onnx import to_onnx  
    logging.info("Serializing model")  
    onx = to_onnx(clr, X[:1])  
    with open("model.onnx", "wb") as f:  
        f.write(onx.SerializeToString())  
    logging.info("Model serialized, saved as model.onnx")
```

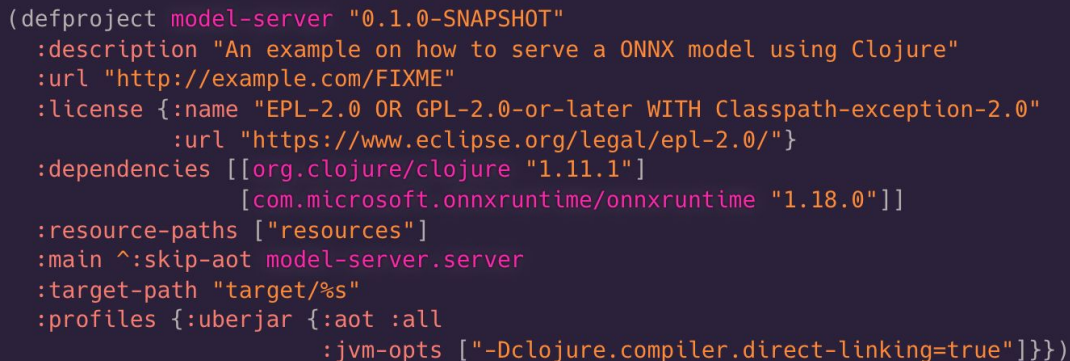
How to use?

Serialization



How to use?

ONNXRUNTIME



```
(defproject model-server "0.1.0-SNAPSHOT"
  :description "An example on how to serve a ONNX model using Clojure"
  :url "http://example.com/FIXME"
  :license {:name "EPL-2.0 OR GPL-2.0-or-later WITH Classpath-exception-2.0"
            :url "https://www.eclipse.org/legal/epl-2.0/" }
  :dependencies [[org.clojure/clojure "1.11.1"]
                 [com.microsoft/onnxruntime/onnxruntime "1.18.0"]]
  :resource-paths ["resources"]
  :main ^:skip-aot model-server.server
  :target-path "target/%s"
  :profiles {:uberjar {:aot :all
                       :jvm-opts ["-Dclojure.compiler.direct-linking=true"]}}})
```

Repo: <https://github.com/caique-lima/onnx-examples>

How to use?

Load the model

```
(ns model-server.server
  (:import [ai.onnxruntime OrtEnvironment])
  (:require [clojure.java.io :as io]))

(def ^:private ort-env
  (OrtEnvironment/getEnvironment))

(defn ^:private ort-session
  [ort-env onnx-file]
  (.createSession ort-env onnx-file))

(defn ^:private to-ort-session
  [artifact-path]
  (ort-session ort-env artifact-path))

(def ^:private ort
  (to-ort-session (.getPath (io/resource "model.onnx"))))
```

Repo: <https://github.com/caique-lima/onnx-examples>

How to use?

Transform the inputs

```
(ns model-server.adapter.onnx
  (:import [ai.onnxruntime OnnxTensor]
           [java.nio FloatBuffer]))

(defn ^:private create-float-buffer [data]
  (let [size (* (count data) (count (first data)))
        buffer (FloatBuffer/allocate size)]
    (doseq [row data]
      (doseq [item row]
        (.put buffer (float item))))
    (.flip buffer)
    buffer))

(defn input->onnx-tensor
  [ort-env input]
  (let [input-size (count input)
        input-row-size (count (first input))]
    (OnnxTensor/createTensor ort-env (create-float-buffer input) (long-array [input-size input-row-size]))))
```

How to use?

Inference



```
(ns model-server.controller.inference)
```

```
(defn infer  
  [input ort]  
  (.run ort {"X" input}))
```

Benchmark

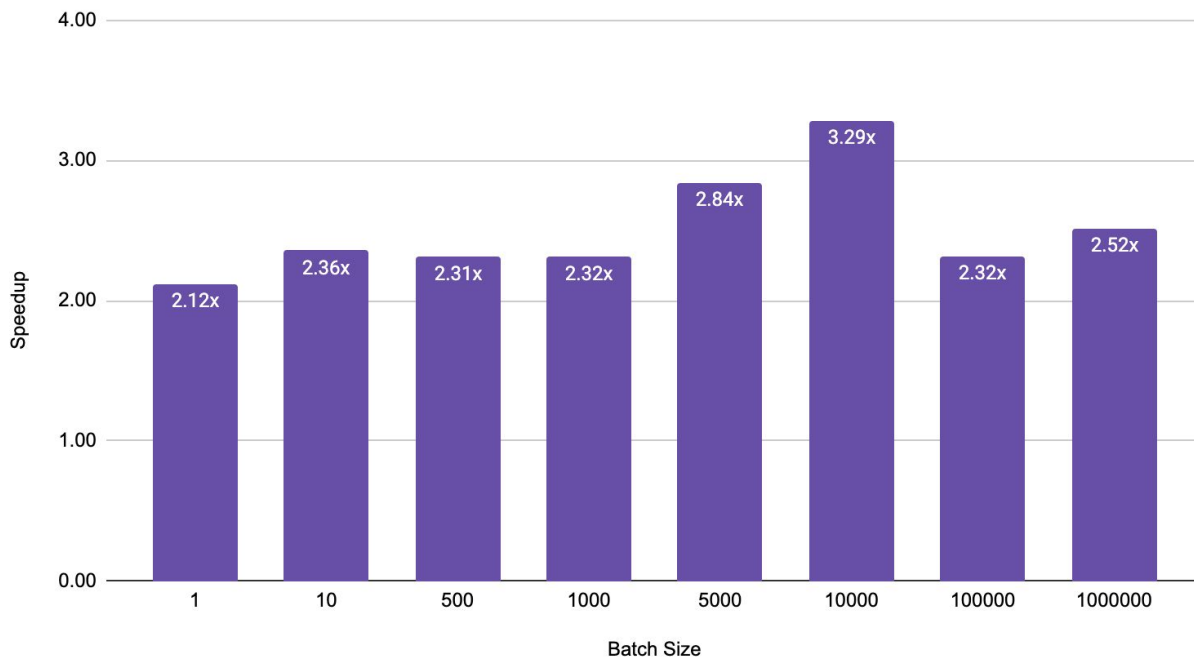
Test design

- We used a Logistic Regression model with 10 features, based on diabetes dataset
- Exposed the model as a Rest API
- Apache Jmeter to perform the requests
- Comparison is done only on time to run the inference
- Data was randomly generated during the tests
- We compared the p90 inference time between the two solutions

Benchmark

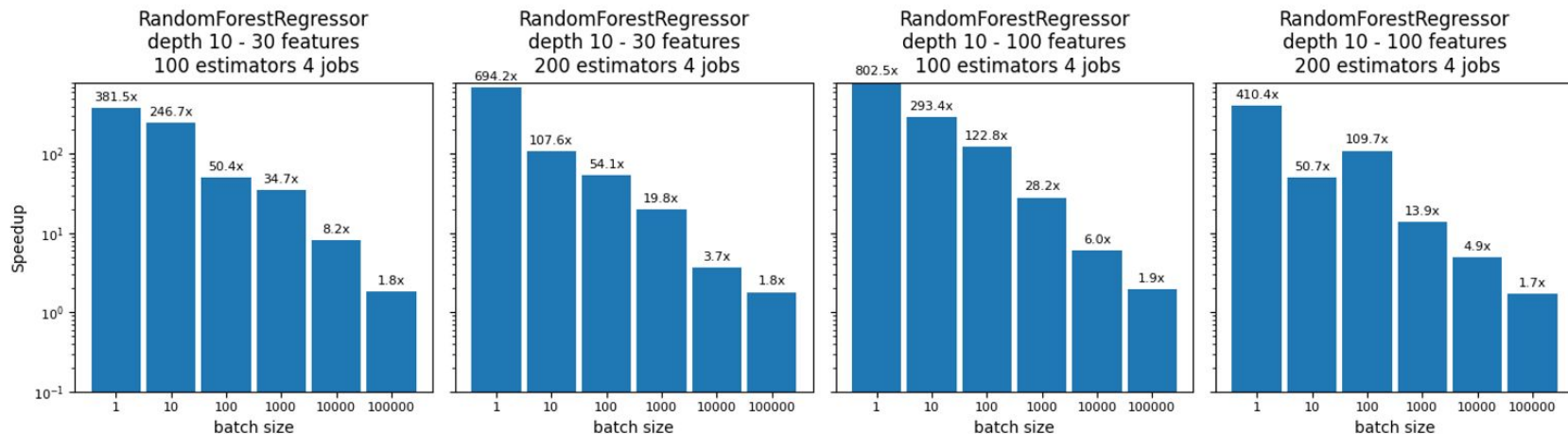
Results

LinearRegression - 10 features - Clojure ONNX vs. Scikit speedup



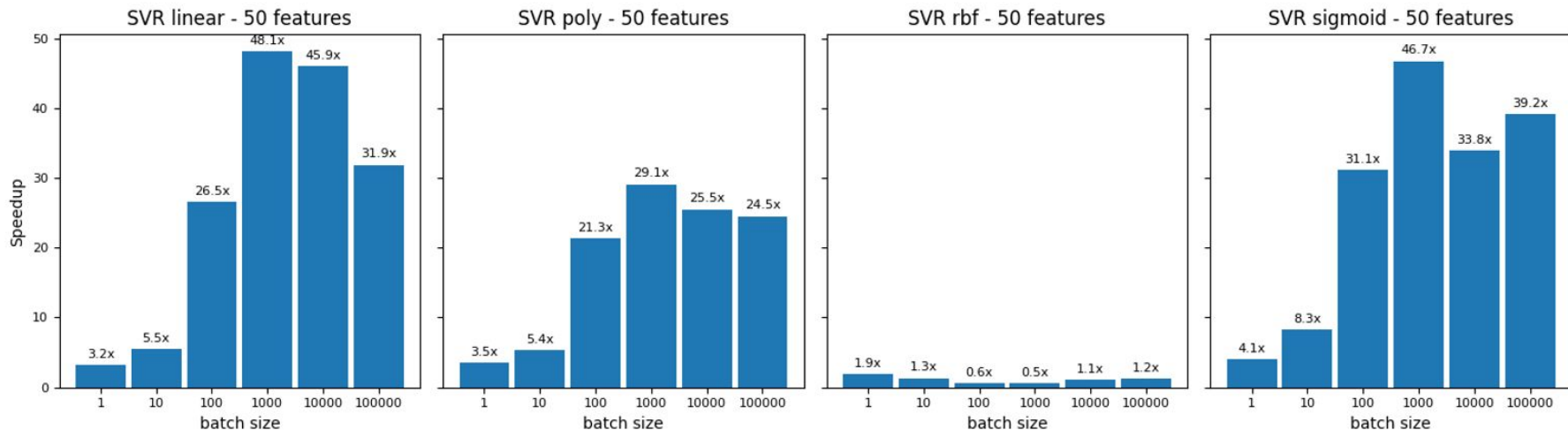
Benchmark

Public benchmarks



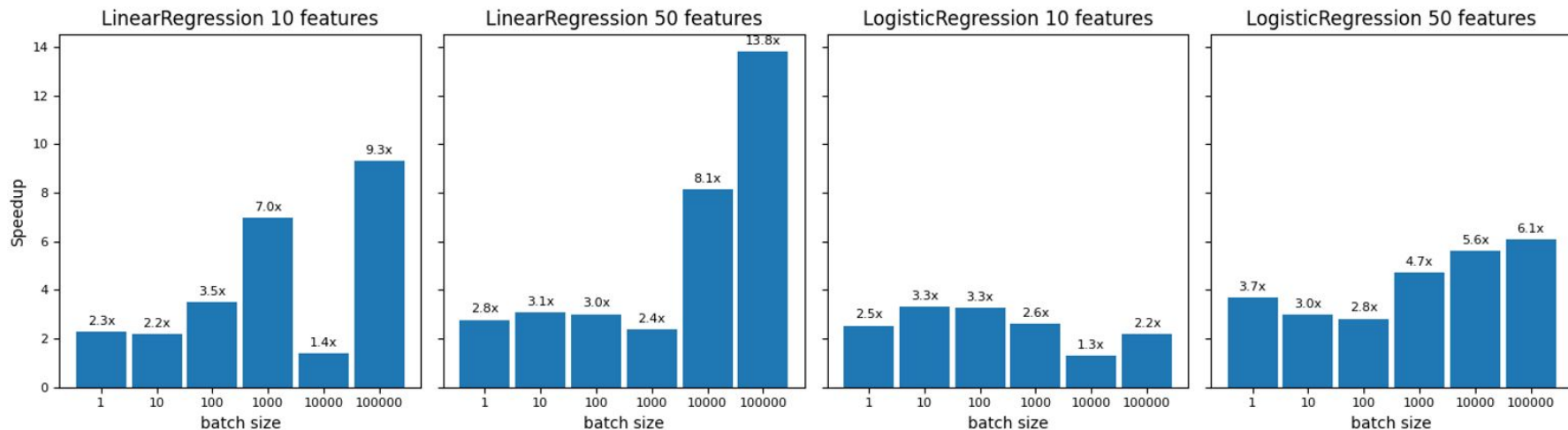
Benchmark

Public benchmarks



Benchmark

Public benchmarks



Why we are not using it?

Why we are not using it?

Our challenges

Feature engineering

Data scientists are accustomed to creating the model and data transformations in a single pipeline. This makes it hard to decouple what constitutes the model itself from the data transformations.

Legacy code

The project to migrate our core models to ONNX would involve high risks and costs, so we ended up keeping them running in Python.

Business value

Is often pretty hard for the business to value the long term approach in favor to keep moving fast on top of existing platform.