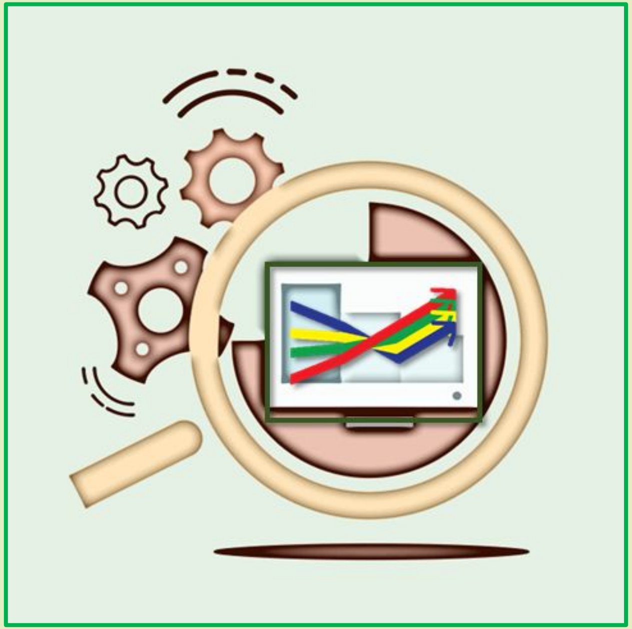
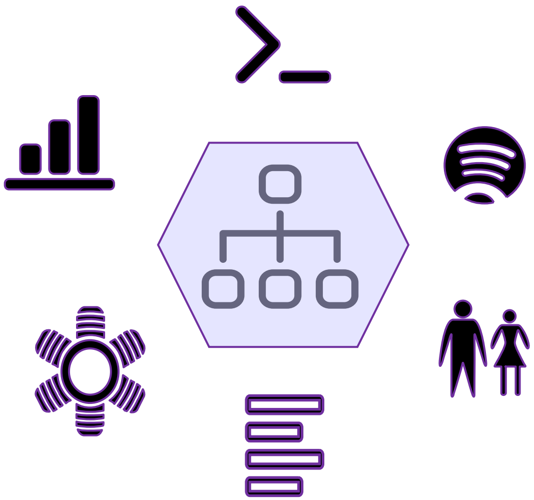


Architecture Approaches for ML Systems

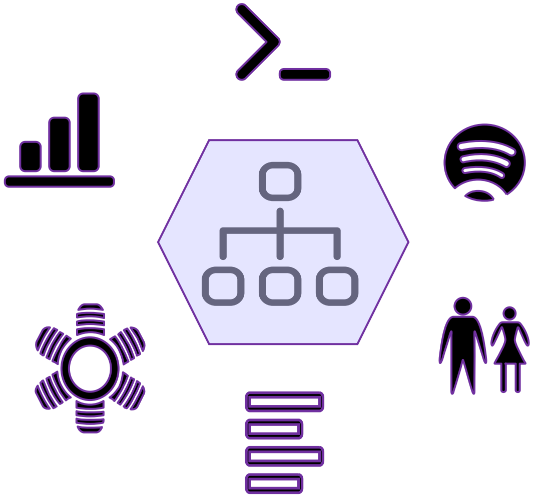
Train In Data

ML System Architectures



1. Model embedded in application
2. Served via a dedicated service
3. Model published as data (streaming)
4. Batch prediction (offline process)

Serving ML Models - Formats



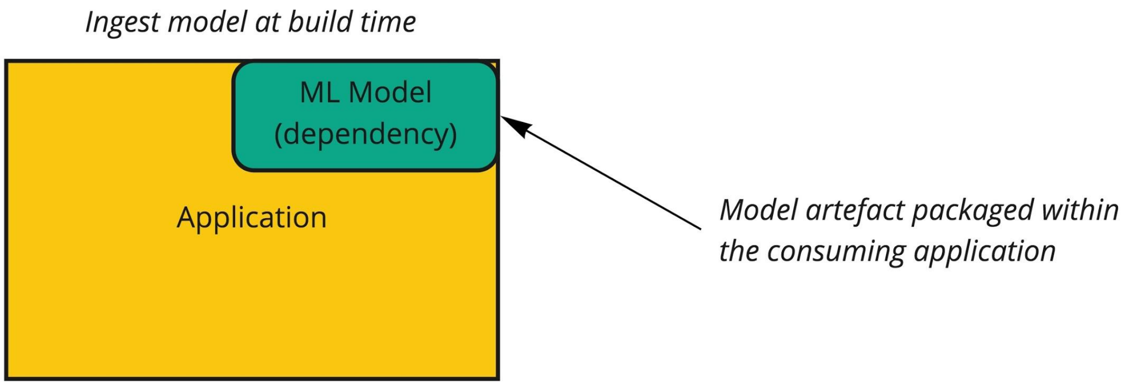
- Serializing the model object with pickle.
- MLFlow (MLeap module) provides a common serialization format for exporting/importing Spark, Scikit-learn, and Tensorflow models.
- Language-agnostic exchange formats to share models, such as PMML, PFA, and ONNX.

Architecture 1: Embedded

Pre-Trained: Yes

Predict-on-the-fly: Yes

Variations: Embedded on mobile device (e.g. Core ML), running in the browser (Tensorflow.js)

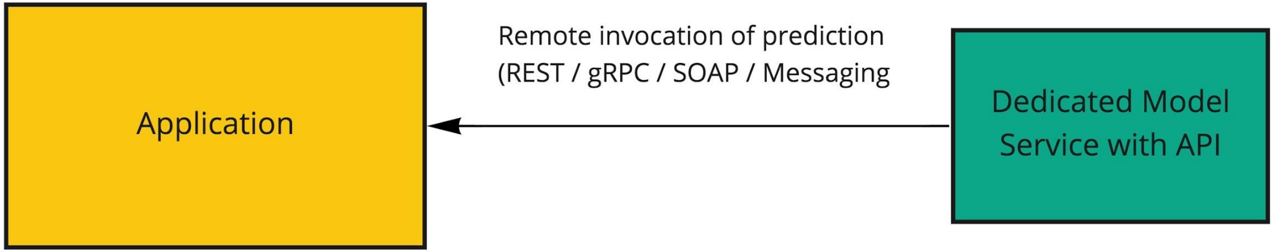


Architecture 2: Dedicated Model API

Pre-Trained: Yes

Predict-on-the-fly: Yes

Variations: Many. See also Architecture 3



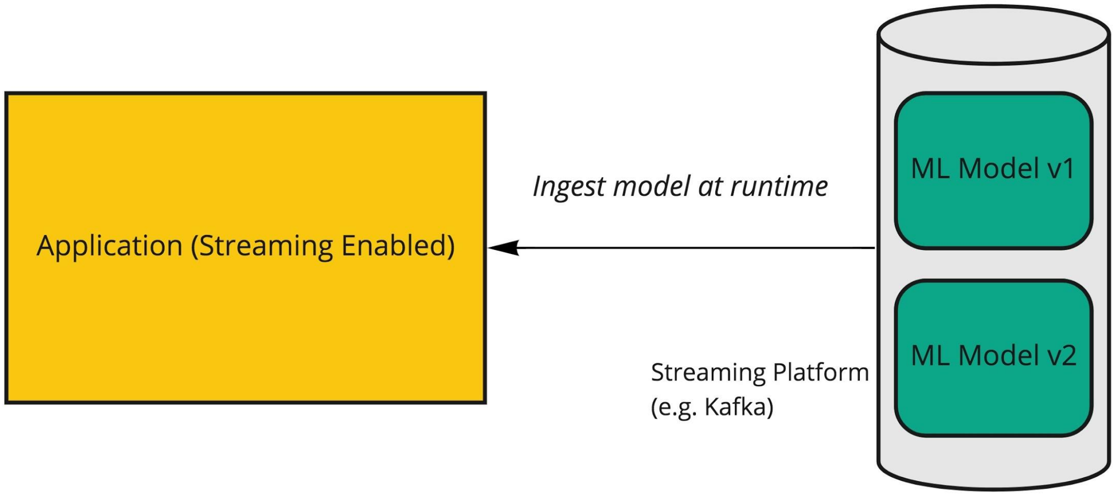
Model is wrapped in a service that can be deployed independently

Architecture 3: Model Published as Data

Pre-Trained: Yes

Predict-on-the-fly: Yes

Variations: Different publish/subscribe patterns



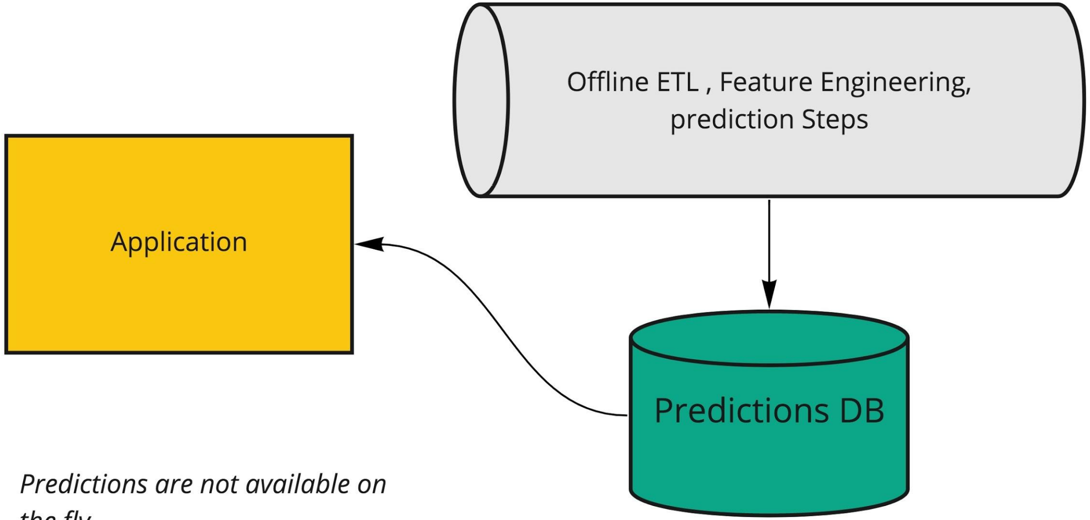
Application subscribes to events and ingests new models in memory

Architecture 4: Offline Predictions

Pre-Trained: Yes

Predict-on-the-fly: No

Variations: Serve predictions via API, CSV, dashboards



Predictions are not available on the fly

Architecture Comparison

	Pattern 1 (Embedded)	Pattern 2 (API)	Pattern 3 (Streaming)	Pattern 4 (Offline)
Prediction	On the fly	On the fly	On the fly	Batch Offline
Prediction result delivery	Within app process	Via API	Streaming via Message Queue	Shared DB, API, file
Latency for prediction	Low	Moderate	Depends	Hours/Days
System Management Difficulty	So so	Easy	Very Hard	So so
Model Update requires deployment?	Yes	Yes (of model service)	No	Yes

(1) and (2) are the focus of this course