UNIVERSITÀ DEGLI STUDI DI TRIESTE

Class 6: Decision-Making and Interfaces for ML Models

Master Course:

Data-driven Systems Engineering (ML Operations)

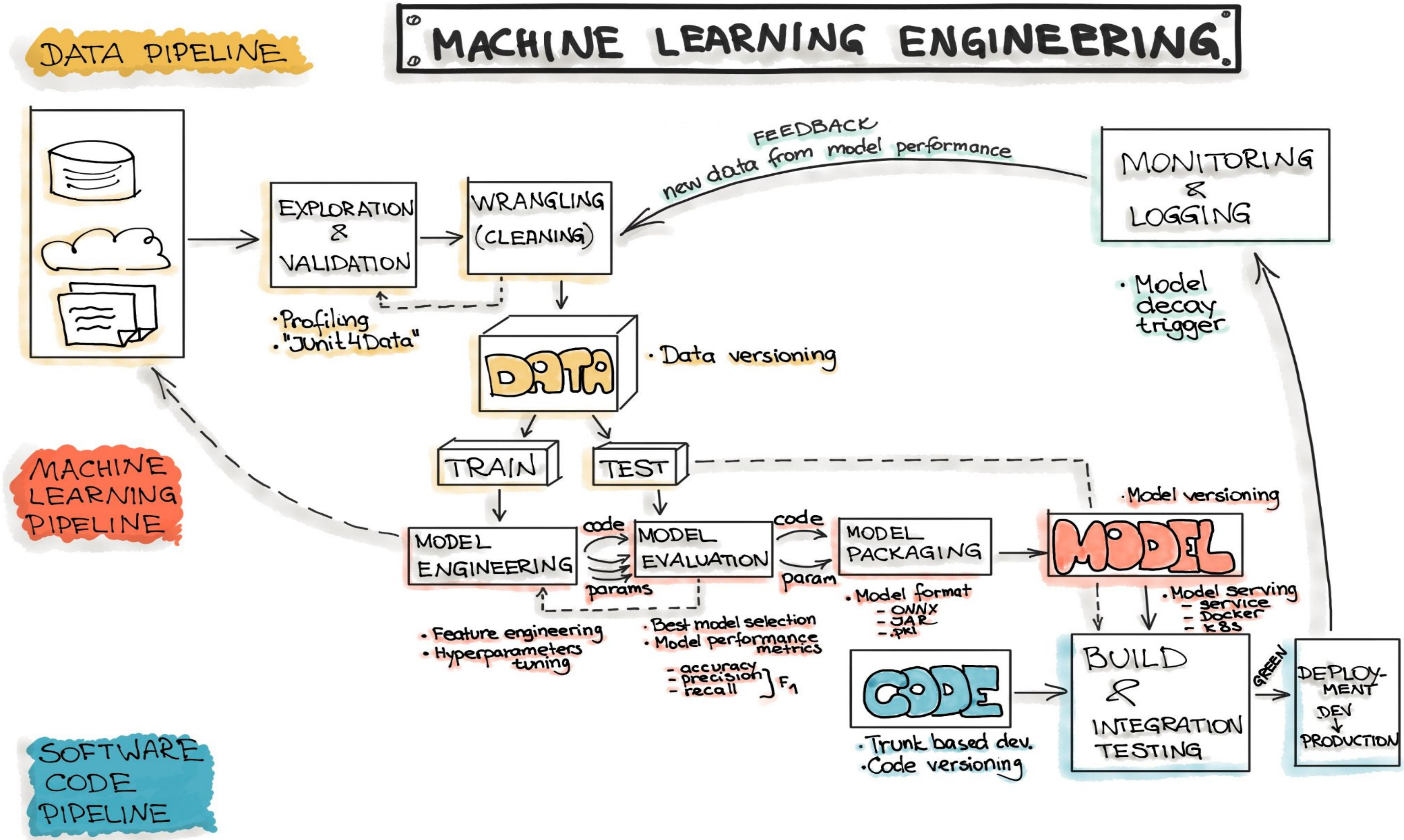
440MI and 305SM

Agenda

- Learning Goals
- From Models to Decisions
- Visualizing Model Predictions
- Interfaces for Machine Learning
- Introduction to Streamlit
- Live Demo: Building a Streamlit
- Discussion and Best Practices

Learning Goals

- Understand how predictions from ML models can support decision-making.
- Visualize model outputs interactively.
- Build simple web interfaces for ML using Streamlit.
- Translate technical predictions into actionable insights.


MACHINE LEARNING ENGINEERING
DATA PIPELINE
MACHINE LEARNING PIPELINE
SOFTWARE CODE PIPELINE
EXPLORATION & VALIDATION
WRANGLING (CLEANING)
·Profiling
·"JUnit4Data"
FEEDBACK
new data from model performance
MONITORING & LOGGING
·Model decay trigger
DATA
·Data versioning
TRAIN
TEST
MODEL ENGINEERING
code
params
MODEL EVALUATION
code
param
MODEL PACKAGING
MODEL
·Model versioning
·Feature engineering
·Hyperparameters tuning
·Best model selection
·Model performance metrics
- accuracy
- precision
- recall
F1
·Model format
- ONNX
- JAR
- .pkl
·Model serving
- service
- Docker
- K8S
CODE
·Trunk based dev.
·Code versioning
BUILD & INTEGRATION TESTING
GREEN
DEPLOYMENT
DEV
PRODUCTION

From Models to Decisions

A machine learning model is only useful when its output is **interpreted and acted upon**.
For example:

- A **fraud detection model** triggers a *review process*.
- A **credit risk model** informs *loan approval*.
- A **classification model** in healthcare supports *diagnosis decisions*.

Open points:

- Predictions are not final decisions — they are **decision-support tools**.
- Importance of **thresholds**, **confidence**, and **cost of false decisions**.
- Role of **human-in-the-loop** systems.

Visualizing Model Predictions

- Prediction Distribution
  a. Show predicted probabilities (e.g., fraud probability, disease likelihood).
  b. Use histograms, calibration plots, or scatter plots.
- Confusion Matrix and Thresholds
  a. Demonstrate how different thresholds affect outcomes.
  b. Visualization idea: a slider that adjusts decision threshold dynamically.
- Feature Importance
  a. Visualize feature_importances_ (tree models) or SHAP values.
  b. Help users interpret why a prediction was made.

Interfaces for Machine Learning

Objective: Enable interaction between users and models.

Common interface types:

Type	Description	Example Tools
Web Apps	Interactive apps for model exploration	Streamlit, Gradio, Dash
Dashboards	Static or semi-interactive results	Power BI, Tableau
APIs	Backend-only services	Flask, FastAPI
Embedded Interfaces	Integration in other software	ERP systems, Chatbots

- Improves accessibility of ML results.
- Encourages collaboration between data scientists and stakeholders.
- Provides real-time feedback on model performance.

Introduction to Streamlit

What is Streamlit?

A Python-based open-source framework for creating interactive data apps and dashboards quickly.

Core Features:

- Build interfaces with pure Python (no HTML/JS).
- Supports interactive widgets (sliders, text boxes, buttons).
- Ideal for model demos, dashboards, and educational apps.

Installation:

```
pip install streamlit
```

Run an app:

```
streamlit run app.py
```

Streamlit

Live Demo: ML Decision Interface

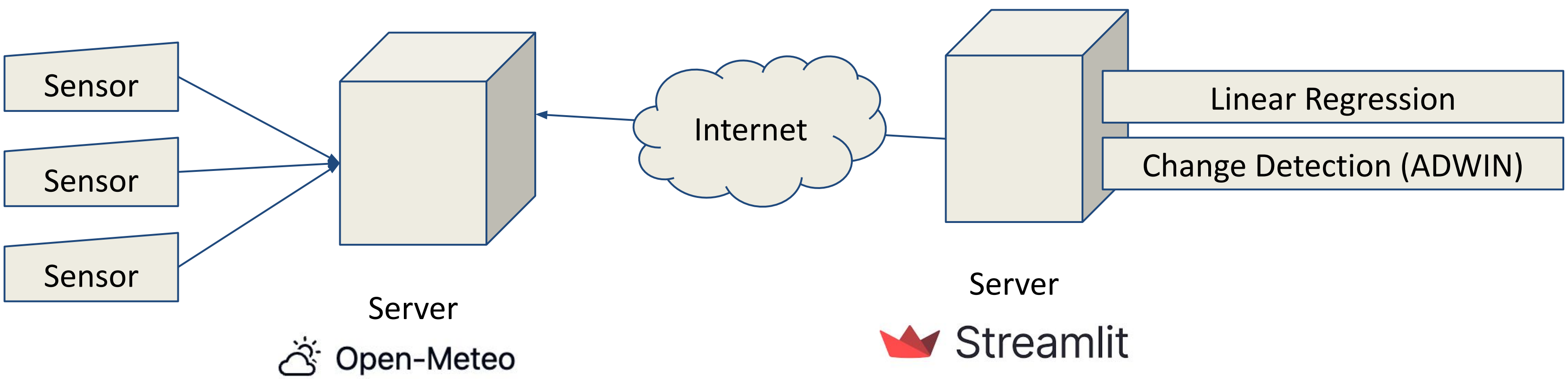
Streamlit Example Code

```
import streamlit as st
import pickle
import numpy as np

# Load model
model = pickle.load(open('model.pkl', 'rb'))

st.title("Credit Card Fraud Detection App")

st.sidebar.header("Transaction Input Features")
amount = st.sidebar.number_input("Transaction Amount (€)", min_value=0.0)
v1 = st.sidebar.slider("Behavior Score (V1)", -5.0, 5.0, 0.0)
v2 = st.sidebar.slider("Risk Factor (V2)", -5.0, 5.0, 0.0)

input_data = np.array([[v1, v2, amount]])
prediction = model.predict(input_data)[0]
prob = model.predict_proba(input_data)[0][1]

st.subheader("Prediction Result")
if prediction == 1:
    st.error(f"⚠️ Fraud detected (probability: {prob:.2f})")
else:
    st.success(f"✅ Legitimate transaction (probability: {prob:.2f})")
```

Project:

Online Weather Prediction with Streamlit and River

Demonstrating Real-Time Machine Learning and Concept Drift Detection



Project: Online Weather Prediction with Streamlit and River

Traditional ML: ML is the process of training a model from a fixed dataset to make predictions or decisions on new, unseen data.

Characteristics:

- Works with a **static** dataset.
- Model **trained once**, then deployed.
- Learning occurs **offline** (no live updates).
- Training is usually **computationally intensive**.

Online ML: Online ML trains models incrementally, one instance (or mini-batch) at a time, updating knowledge continuously as new data arrives.

- Updates model weights for **each observation** (Incremental Learning).
- Keeps **minimal statistics**, not full data (Memory Efficient).
- **Adjusts to data drift** or changing patterns (Adaptive).
- Suitable for **continuous streams and IoT** (Real-Time Prediction).

Project: Online Weather Prediction with Streamlit and River

Stream Mining is the process of extracting knowledge and detecting patterns from **unbounded, fast, and evolving data streams** in real time. Includes:

- **classification**
- **clustering**
- **anomaly detection**
- **drift detection**

Goal:
Maintain **current knowledge** about a dynamic process while data flows continuously.

Property	Description
Unbounded data	Stream never stops — must process data once (“single-pass”).
Time constraints	Must produce results within milliseconds.
Concept drift	Patterns evolve — model must adapt automatically.
Limited memory	Cannot store the full stream.

Project: Online Weather Prediction with Streamlit and River

Aspect	Data Mining	Stream Mining
Data Nature	Works on **static, finite datasets** stored in databases or files.	Works on **continuous, unbounded data streams** that arrive in real time.
Data Access	Full dataset is **available in advance**.	Data arrives **sequentially**, often processed **once** (single-pass).
Learning Mode	**Offline / batch learning** — model trained once on complete data.	**Online / incremental learning** — model updated continuously as new data arrives.
Storage	Requires **large storage** to keep all data accessible.	Operates with **limited memory**, storing only summaries or windows of data.
Processing Strategy	Allows **multiple scans** over the dataset.	Only **one scan** (or very few) — cannot revisit past data.
Computation	Time-insensitive; focuses on **accuracy** and **completeness**.	Time-sensitive; focuses on **speed** and **adaptivity**.
Data Distribution	Assumes **stationary distribution** (data doesn't change over time).	Handles **non-stationary distribution** (concept drift, evolving patterns).
Algorithm Examples	Decision Trees (CART, C4.5), k-Means, Apriori, Random Forest.	Hoeffding Trees, CluStream, ADWIN, Online SGD, DenStream.
Applications	Customer segmentation, market basket analysis, historical analytics.	Network intrusion detection, IoT sensor monitoring, stock market prediction, real-time anomaly detection.
Model Adaptation	Retraining required when new data becomes available.	Automatically adapts with **incremental updates**.
Output Type	Static models or reports (batch results).	Continuous predictions, evolving summaries, real-time alerts.
Goal	Discover hidden patterns from **historical data**.	Continuously extract actionable knowledge from **dynamic data streams**.

Project: Online Weather Prediction with Streamlit and River

Stochastic Gradient Descent (SGD) is an iterative optimization algorithm used to minimize a model's loss function by updating weights incrementally after each training example.

In the online learning context, SGD updates happen one observation at a time, enabling real-time model adaptation.

For each new data point (x_t, y_t):

$$\hat{y}_t = w \cdot x_t$$

$$w_{t+1} = w_t - \eta \cdot \frac{\partial L(y_t, \hat{y}_t)}{\partial w_t}$$

Where:

- w_t: current model parameters
- η: learning rate (step size)
- L: loss function (e.g., mean squared error)

Project: Online Weather Prediction with Streamlit and River

ADWIN (ADaptive WINdowing) is an algorithm used to detect concept drift — situations where the statistical properties of data change over time.

In streaming scenarios (like weather prediction), ADWIN continuously monitors the **error rate or data distribution**, and signals **drift** when a significant change is detected. Main idea (**sliding window**):

1. When new data arrives, it's added to the window.
2. The algorithm compares **two sub-windows (W_1 and W_2)** within the current window.
3. If their averages differ significantly (using Hoeffding's bound), **drift is declared**.
4. The older portion of the window is then **dropped**, keeping only the most relevant recent data.

If:

$$|\mu_1 - \mu_2| > \varepsilon$$

then drift is detected, where:

$$\varepsilon = \sqrt{\frac{1}{2m} \ln\left(\frac{4}{\delta}\right)}$$

- μ_1, μ_2: means of the two sub-windows
- m: harmonic mean of their sizes
- δ: confidence parameter

Project: **Online Weather Prediction with Streamlit and River**

- Traditional ML models are *trained offline* — they can't adapt quickly to new data.
- In real environments (weather, finance, IoT), **data arrives continuously**.
- Models must **learn incrementally** and detect **changes in patterns** (*concept drift*).

Goal:
Predict temperature in real time using **wind speed** data, continuously updating the model as new data arrives.

Project: Online Weather Prediction with Streamlit and River

Component	Description
Data Source	Open-Meteo API provides live weather data (temperature, wind speed).
Model	Online Linear Regression from River (`LinearRegression + StandardScaler`).
Metrics	MAE (Mean Absolute Error) for performance tracking.
Drift Detection	ADWIN algorithm detects changes in error distribution.
Interface	Streamlit app for real-time visualization and control.

*Stochastic Gradient Descent (SGD)

References:

- River Documentation – https://riverml.xyz
- Streamlit Documentation – https://streamlit.io
- Open-Meteo API – https://open-meteo.com
- Bifet, A., & Gavaldà, R. (2007). *Learning from Time-Changing Data with Adaptive Windowing (ADWIN).*
- Grus, J. (2023). *Data Science from Scratch.* O'Reilly Media.