UNIVERSITÀ DEGLI STUDI DI TRIESTE

Class 5: Python ML Project Modeling

Master Course:

Data-driven Systems Engineering (ML Operations)

440MI and 305SM

Agenda

- Learning Goals
- Definitions
- RICE Framework
- Computational Representation
- Modeling and Abstraction
- Why Model Adaptation Matters?
- Why Continuous Learning is Needed?
- Drift Detection Mechanisms
- Real-World!

Learning Goals

- Understand ML modeling paradigms
- Explore online adaptation
- Learn how to expose and serve ML models
- Examine ML ecosystems and monitoring

Definition: What is Modeling?

- In machine learning, modeling refers to the formalization of data patterns through computational representations, encoding empirical relationships found in data into a mathematical or algorithmic form that can generalize beyond observed examples.

$$f: X \rightarrow Y$$

RICE Framework in Machine Learning

RICE = Representation–Information–Computation–Evaluation

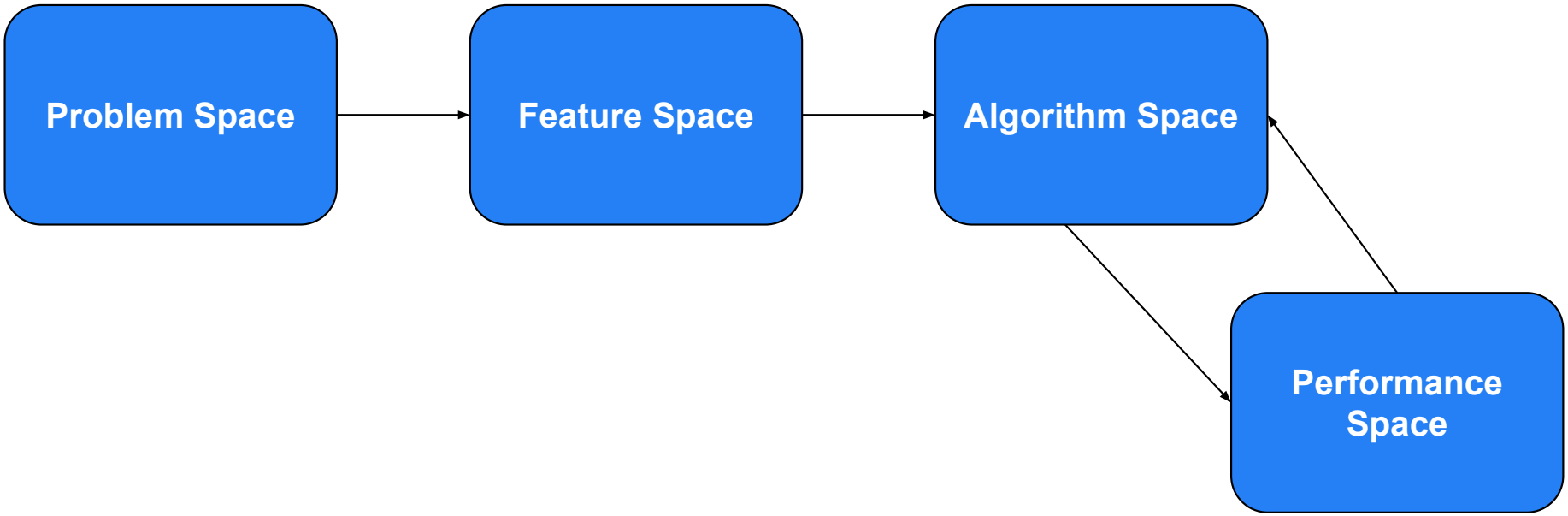
Defines the complete cycle of ML system design and optimization:

- *Representation:* how knowledge is structured
- *Information:* what data is available and its quality
- *Computation:* how learning occurs algorithmically
- *Evaluation:* how performance is measured and improved

Together, RICE forms the foundation for iterative, data-driven model development and operational excellence. *(Domingos, 2012; Mitchell, 1997; Jordan & Mitchell, 2015)*

RICE Framework in Machine Learning

The RICE framework thus offers an integrative view of **machine learning as an iterative optimization process across four interdependent spaces**, aligning closely with both theoretical and applied ML system design.



Computational Representation

Models differ in how they represent and compute this mapping:

Model Type	Representation Mechanism	Example
Linear Models	Parameter vector	Linear Regression
Decision Trees	Hierarchical partitioning of feature space	CART, Random Forest
Probabilistic Models	Conditional probability distributions	Naïve Bayes, HMM
Neural Networks	Nonlinear composition of layers	CNNs, RNNs, Transformers

Each of these representations defines a *different inductive bias* — the assumptions that guide learning from limited data (Mitchell, 1997).

Modeling as Abstraction

From a systems engineering perspective, modeling is a process of **abstraction**:

- Reducing real-world complexity into computationally tractable structures.
- Capturing relevant signal while filtering out noise.
- Enabling automation, prediction, and decision-making at scale.

As emphasized by Bishop (2006), the ultimate goal is not merely fitting data but learning *generalizable representations* that can make accurate predictions on unseen cases.

Modeling as Abstraction

Supervised Learning:

- Learn from labeled data.
- Tasks: classification, regression.
- Algorithms: Linear Regression, Random Forest, SVM, Neural Networks.
- High accuracy with sufficient labels.
- Costly labeling process, overfitting risk

Domain	Application	Typical Algorithm	Rationale
Finance	Credit scoring, fraud detection	Random Forest, XGBoost	Robust to heterogeneous data, interpretable feature importance
Healthcare	Medical diagnosis	Neural Networks, SVM	Can model non-linear feature interactions
Manufacturing	Predictive maintenance	Gradient Boosting, RF	Handles tabular, sensor-based data
NLP / Vision	Image classification, sentiment analysis	Deep CNNs, Transformers	Exploit large labeled datasets and hierarchical features

Modeling as Abstraction

Unsupervised Learning:

- Discover hidden structures in unlabeled data.
- Algorithms: K-Means, DBSCAN, PCA, Autoencoders.
- Use: clustering, anomaly detection.
- Reveals hidden patterns.
- Interpretation difficulty, hyperparameter sensitivity.

Domain	Application	Method	Motivation
Customer Analytics	Market segmentation	K-Means / GMM	Identify behavioral clusters
IoT / Sensor Data	Anomaly detection	Autoencoders	Learn normal patterns, detect deviations
Text Mining	Topic modeling	Latent Dirichlet Allocation	Discover latent topics
Bioinformatics	Gene expression grouping	Hierarchical clustering	Reveal biological pathways

Modeling as Abstraction

Semi-supervised Learning:

- Mix of labeled and unlabeled data.
- Techniques: self-training, co-training, graph-based SSL.
- Challenges: validation on partially labeled data.
- Example: label propagation in text classification.

Domain	Application	Strategy	Benefit
Medical Imaging	Disease detection	Pseudo-labeling + CNN	Labeled data limited; unlabeled images abundant
Cybersecurity	Threat classification	Graph SSL	Dynamic patterns with few labels
Autonomous Driving	Object detection	Self-training	Leverage continuous sensor data
NLP	Intent classification	MixMatch	Augment sparse annotations

Why Model Adaptation Matters?

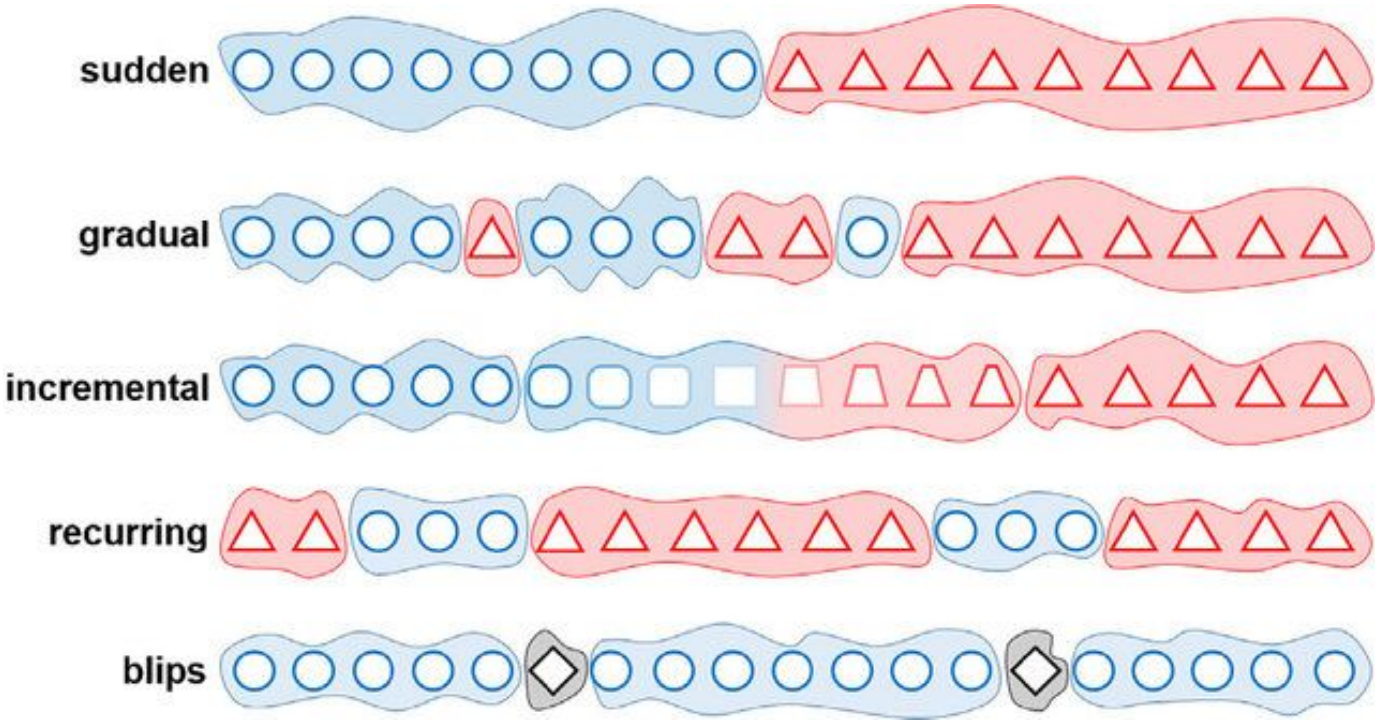
Concept drift occurs when the statistical properties of the target variable or data distribution change over time, causing the predictive relationship P(Y | X) to evolve.

In dynamic environments, the model's learned function f:X→Yf: X \rightarrow Yf:X→Y no longer reflects current reality.

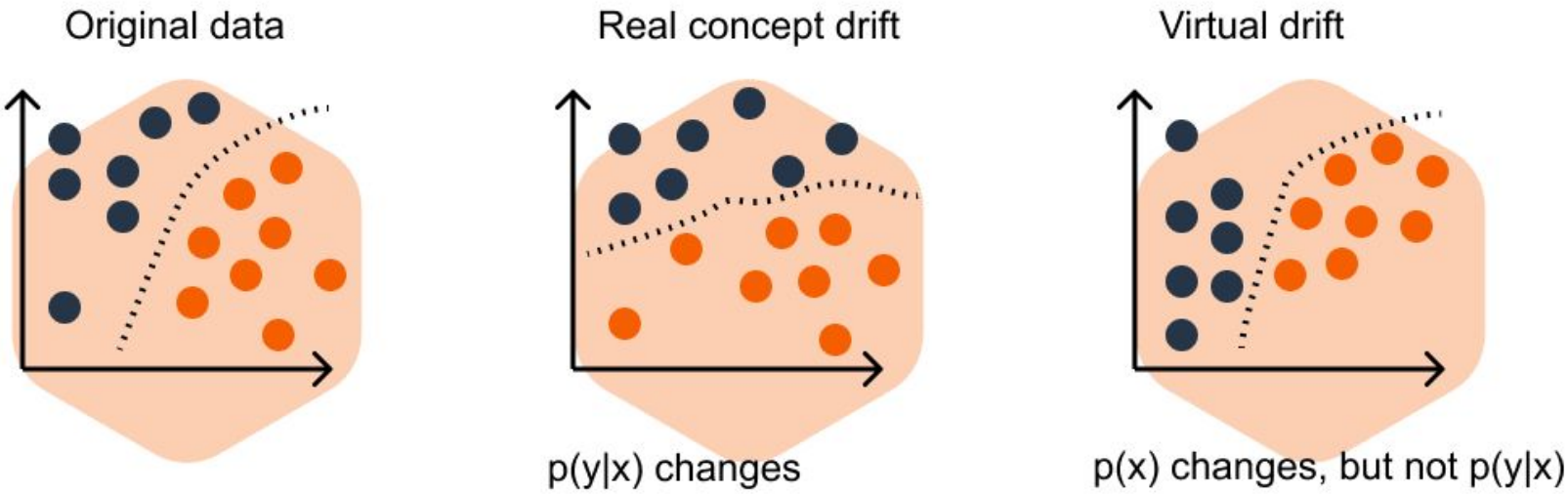
Taxonomia:

Type	Description	Example
Sudden Drift	Abrupt shift in data distribution	New regulation changes transaction patterns
Incremental Drift	Gradual change over time	Customer preferences evolve slowly
Recurring Drift	Past concepts reappear cyclically	Seasonal product demand

Why Model Adaptation Matters?



Why Model Adaptation Matters?



Virtual Drift	Feature distribution changes, but target relation remains	Sensor calibration changes

Why Continuous Learning is Needed?

- Continuous learning = capability to update model incrementally as new data arrives.
- Avoids full retraining (computationally costly).
- Maintains alignment with evolving data distribution.

Techniques:

- **Online Learning Algorithms:** Perceptron, SGD-based (Stochastic Gradient Descent) algorithms
- **Streaming Ensembles:** Hoeffding Trees, Adaptive Random Forest (Bifet et al., 2010).
- **Window-based Adaptation:** ADWIN (Adaptive Windowing).

Online Machine Learning

Example - Very Fast Decision Tree

- A Hoeffding tree (VFDT) is:
  - Incremental Tree,
  - Anytime decision tree induction algorithm
  - Capable of learning from massive data streams, assuming that the distribution generating examples does not change over time.
  - Hoeffding trees exploit the fact that a small sample can often be enough to choose an optimal splitting attribute.
  - Supported mathematically by the Hoeffding bound, which quantifies the number of observations (in our case, examples) needed to estimate some statistics within a prescribed precision.

https://www.youtube.com/watch?v=jDxpm53hJSA

Drift Detection Mechanisms

- **Goal:** Identify when model predictions deviate due to changes in data distribution.
- **Approaches:**
  - *Error-rate monitoring:* Sudden rise in model error → possible drift.
  - *Statistical tests:* Kolmogorov–Smirnov, Page-Hinkley, ADWIN.
  - *Data-driven monitors:* compare distributions of input features or latent embeddings.
- **Adaptive Response:**
  - Retrain model on recent data window.
  - Replace outdated model component.
  - Adjust learning rate dynamically.

Real-world Strategies & Architectures

- **Industry Examples:**
  - *Fraud Detection:* Continuous retraining with rolling window data.
  - *Recommender Systems:* User-item interactions evolve hourly.
  - *Predictive Maintenance:* Sensor patterns drift with machine wear.
- **Engineering Patterns:**
  - Data stream ingestion → preprocessing (Kafka, Flink)
  - Incremental model updates (River, Vowpal Wabbit)
  - Performance monitoring (EvidentlyAI, Prometheus)
  - Automated retraining triggers on drift signals

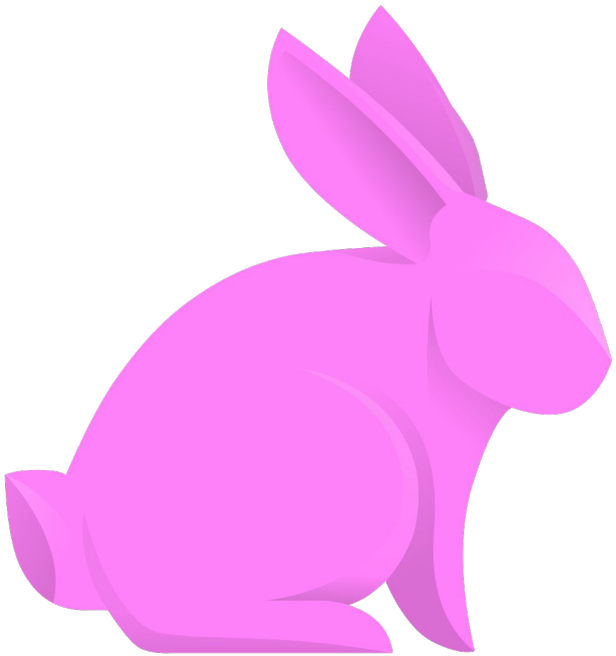
Real-world Strategies & Architectures

- Data stream ingestion → preprocessing (Kafka, Flink)
  - Apache Kafka is a distributed event streaming platform used to build real-time data pipelines;
  - Apache Flink is a unified stream and batch data processing framework for real-time and large-scale data;

Real-world Strategies & Architectures

- Incremental model updates (River, Vowpal Wabbit)

River



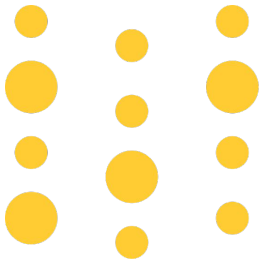
Real-world Strategies & Architectures

- Performance monitoring (EvidentlyAI, Prometheus)

Prometheus

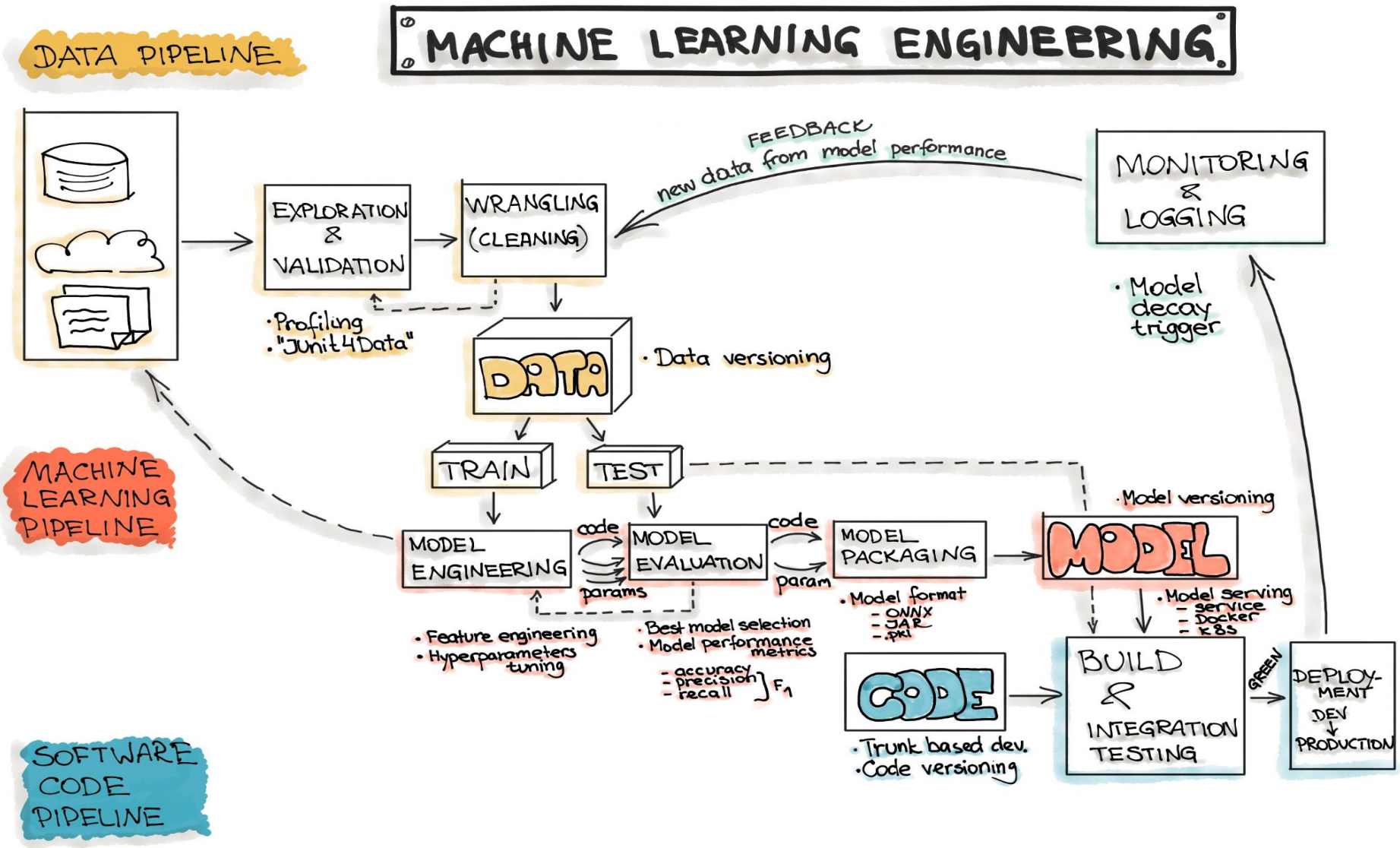
Real-world Strategies & Architectures

- Training



W&B

neptune.ai


MACHINE LEARNING ENGINEERING
DATA PIPELINE
EXPLORATION & VALIDATION
WRANGLING (CLEANING)
FEEDBACK
new data from model performance
MONITORING & LOGGING
· Profiling
· "Junit4Data"
DATA
· Data versioning
· Model decay trigger
MACHINE LEARNING PIPELINE
TRAIN
TEST
· Model versioning
MODEL ENGINEERING
code
params
MODEL EVALUATION
code
param
MODEL PACKAGING
MODEL
· Feature engineering
· Hyperparameters tuning
· Best model selection
· Model performance metrics
- accuracy
- precision
- recall
F1
· Model format
- ONNX
- JAR
- .pkl
· Model serving
- service
- Docker
- k8s
SOFTWARE CODE PIPELINE
CODE
· Trunk based dev.
· Code versioning
BUILD & INTEGRATION TESTING
GREEN
DEPLOY-MENT
DEV
PRODUCTION

References:

- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
- Vapnik, V. N. (1995). The Nature of Statistical Learning Theory. Springer.
- Mitchell, T. M. (1997). Machine Learning. McGraw-Hill Education.
- Domingos, P. (2012). A Few Useful Things to Know About Machine Learning. Communications of the ACM, 55(10), 78–87.
- Jordan, M. I., & Mitchell, T. M. (2015). Machine Learning: Trends, Perspectives, and Prospects. Science, 349(6245), 255–260.