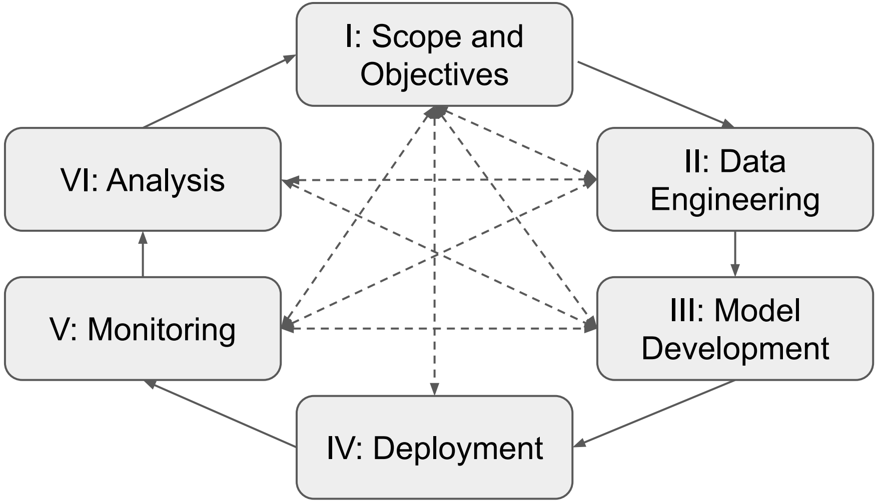
ARiA

Machine-Learning Lifecycle and Workflows

Iterative Stages/Steps for Developing an Operational/Production ML System

Workflows and Metrics Associated with each Stage and *Cutting Across* Multiple Stages

Stages/Steps of the Machine-Learning Lifecycle



Iterative Process:

I. Scope and Objectives
II. Data Engineering
III. Model Development
IV. Deployment
V. Monitoring
VI. Analysis

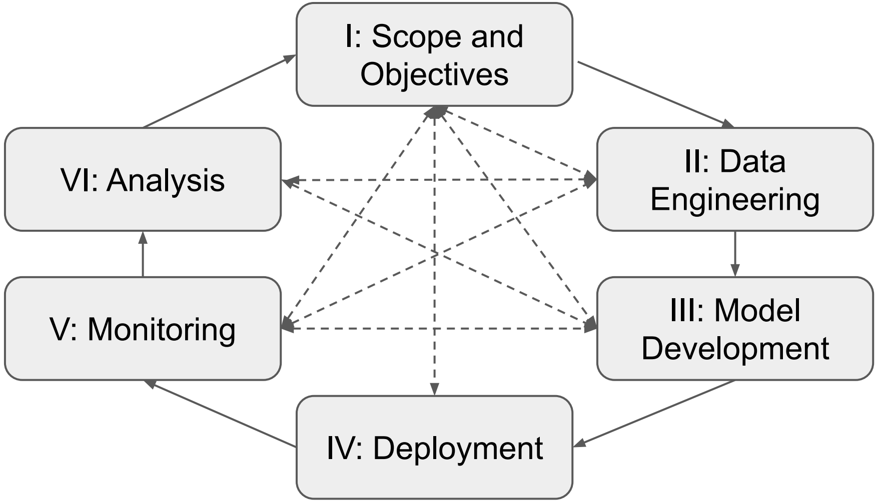
Consensus ML lifecycle derived from

- Chip Huyen, *Designing Machine Learning Systems*, O'Reilly, 2022
- "Well-Architected machine learning lifecycle" https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/well-architected-machine-learning-lifecycle.html accessed 31 JUL 2023 (other elements of this resource, but not this page, cited in original CDAO personna documents)
- H. Veeradhi and K. Abdo, "Your guide to the Red Hat Data Science Model Lifecycle" 09 MAY 2022, https://cloud.redhat.com/blog/your-guide-to-the-red-hat-data-science-model-lifecycle (cited in original CDAO personna documents)

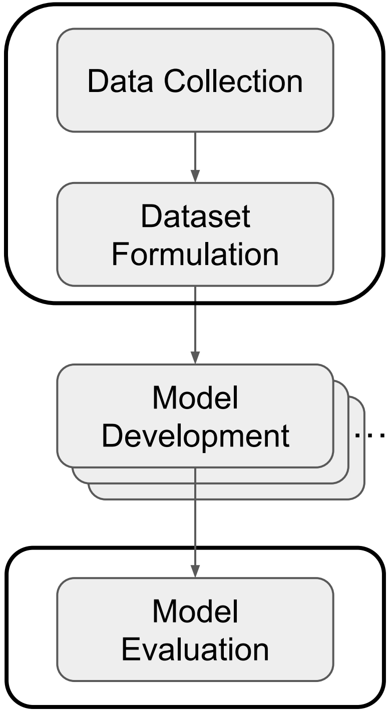
ARiA

Stages/Steps of the Machine-Learning Lifecycle

Operational/Production Lifecycle



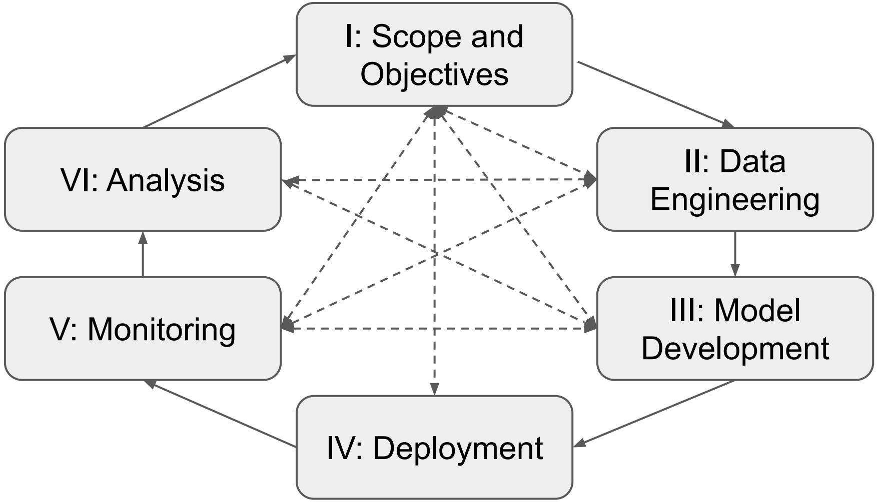
Competition Lifecycle



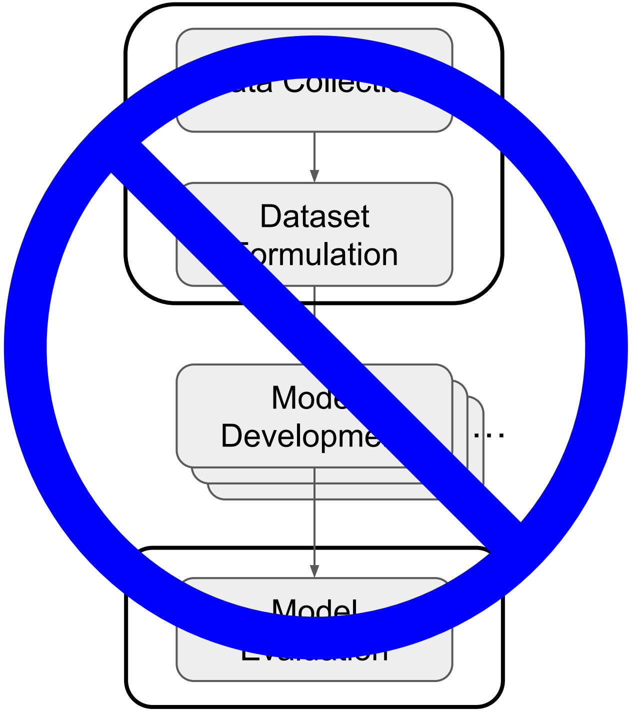
ARiA

Stages/Steps of the Operational Machine-Learning Lifecycle

Operational/Production Lifecycle



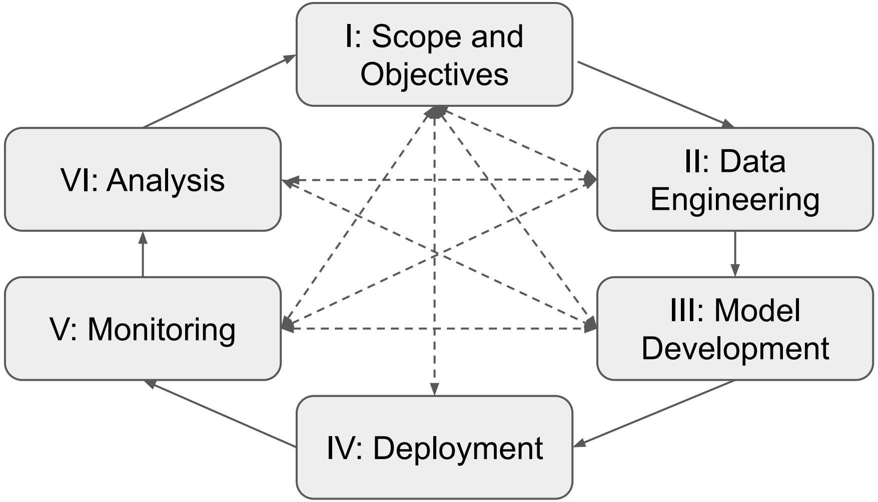
Competition Lifecycle



ARiA

Operational

Stages/Steps of the ^ Machine-Learning Lifecycle



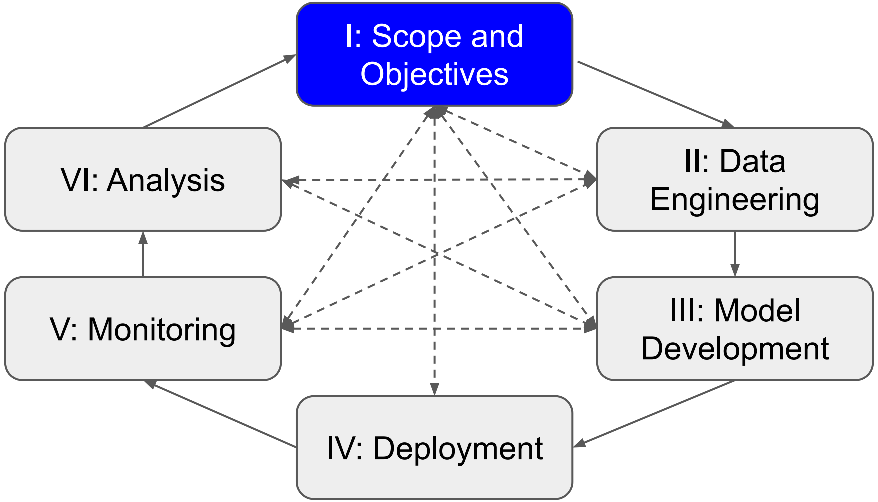
Operational/Production ML System Development:

- not a linear, sequential process
- roles and responsibilities across stages and personnel are dynamic
- “mind” (models) and “data” are both important

The competition is between near-peer nations for *superior operational capability*, *not* between model developers for a *better metric* in a Kaggle-style AI competition

ARiA

Stages/Steps of the Machine-Learning Lifecycle

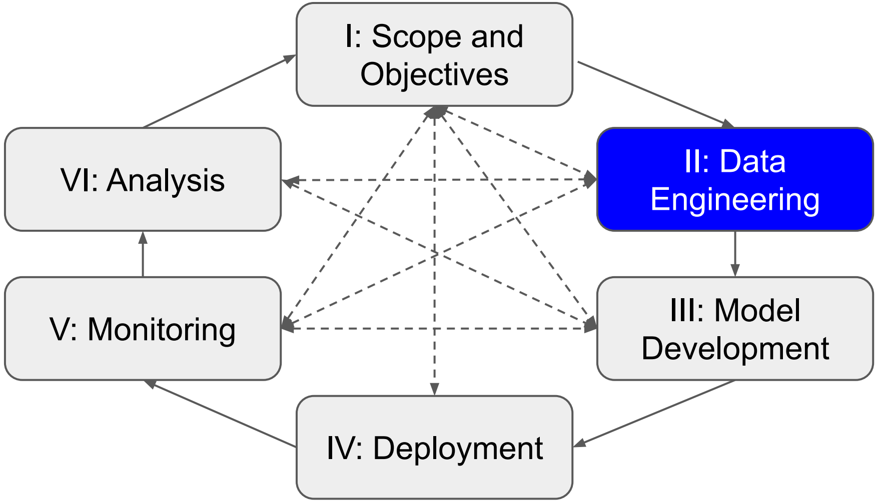


I. Scope and Objectives

- define the scope of the problem and the goals for the solution
- specify operational requirements (performance metrics):
  - materiel release
  - safety analysis
  - doctrine
  - human factors
  - …
- specify operational constraints
  - restrictions on generative factors for evaluating completeness
  - access to labels/groundtruth
  - restrictions on lifetime learning
  - …

ARiA

Stages/Steps of the Machine-Learning Lifecycle

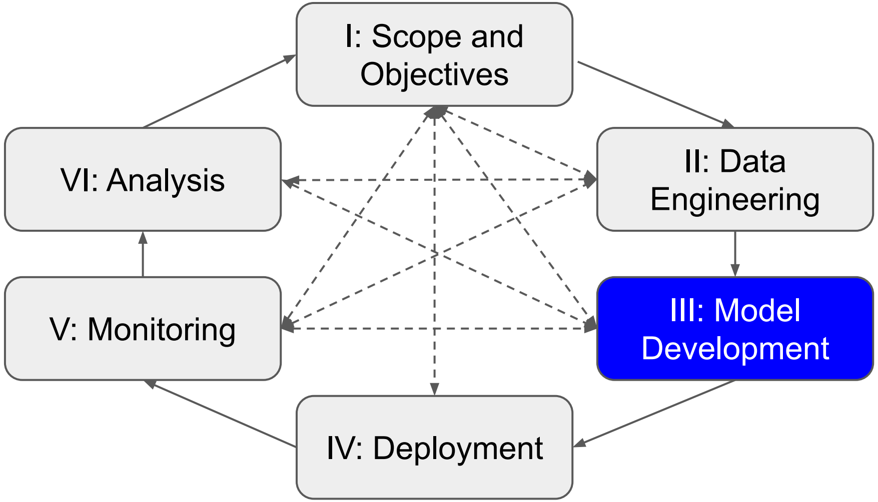


II. Data Engineering

- develop data pipelines
  - data linting
- develop labeling protocols and pipelines
  - label-error detection
- formulate sampling protocols
  - coverage assessment
  - ensure relevance, completeness, balance, and accuracy
- curate static training and test datasets
  - coverage assessment
  - ensure relevance, completeness, balance, and accuracy
  - assess leakage
  - train/test shift
- perform exploratory data analysis
  - data complexity / metafeatures to evaluate achievability of objectives

ARiA

Stages/Steps of the Machine-Learning Lifecycle

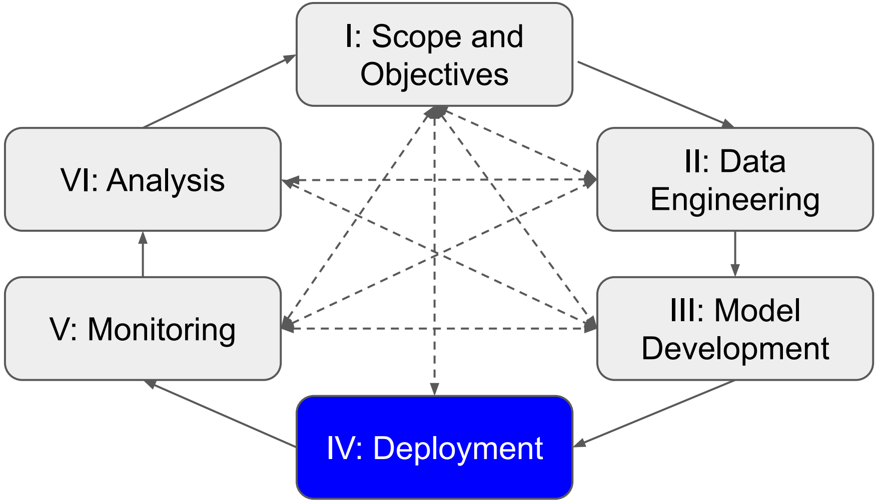


III. Model Development

- model selection
  - metafeatures
  - model/data complexity matching
  - sufficiency assessment
- model training
  - training-data partitioning
  - training-data augmentation
  - leakage, bias, and label errors
- model evaluation
  - performance
  - calibration
  - fairness and generalization
  - robustness and fault tolerance

ARiA

Stages/Steps of the Machine-Learning Lifecycle



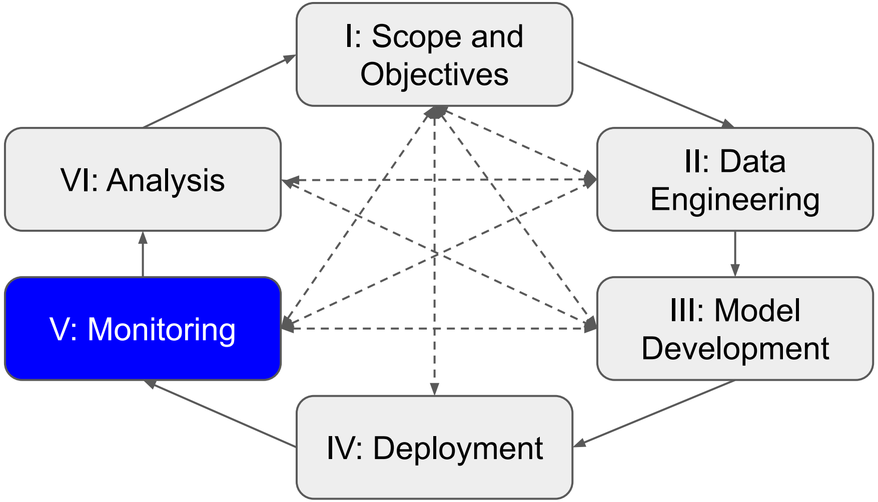
IV. Deployment

- online or batch prediction?
- online or streaming features?
- model update cycle?
- model compression
- model optimization

Deployment decisions can impact model performance metrics and these impacts need to be assessed.

ARiA

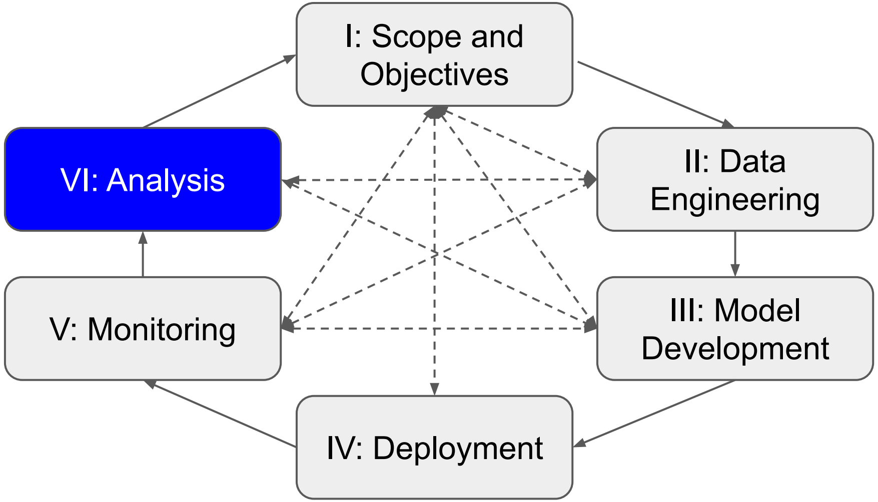
Stages/Steps of the Machine-Learning Lifecycle



V. Monitoring

- data shifts
  - covariate shift - data monitoring
  - label shift - prediction monitoring
  - concept drift
- data monitoring
  - data distribution-shift
- feature monitoring
  - feature distribution-shift
- model monitoring
  - prediction distribution-shift
  - uncertainty/confidence shifts
  - accuracy/performance metrics

Stages/Steps of the Machine-Learning Lifecycle

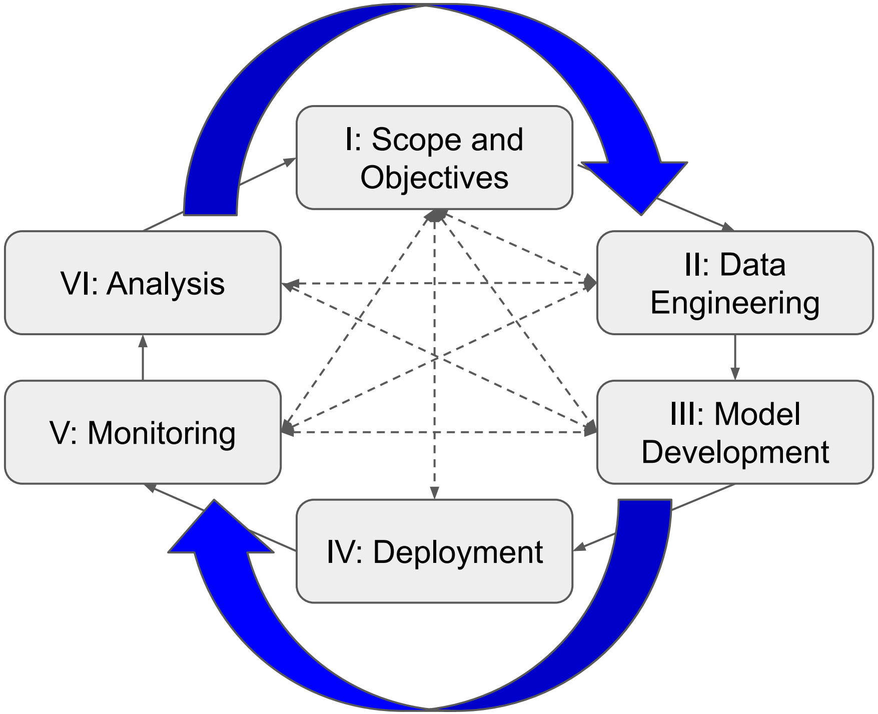


VI. Analysis

- determine whether model achieves specified goal and objective requirements
- refine data engineering and model development stages as needed to achieve objective requirements
- perform analysis on model predictions to generate operational insight that drive refinement of scope and objectives for future iterations

ARiA

Stages/Steps of the Machine-Learning Lifecycle



“Always we begin again”

- developing and deploying an ML system is a never-ending cyclical process
- the world changes and models must change to adapt to the changing world
- modern ML deployment is approaching DevOps timelines
  - Weibo, Alibaba, and ByteDance deploy new ML models on a **10 minute update cycle**
- “People tend to ask me: ‘How often *should* I update my models?’...The right question to ask should be: ‘How often *can* I update my models?’” - Chip Huyen

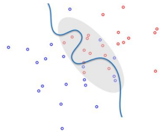
ARiA

Data-Analysis Metrics Library

characterizing image data and its impact on model performance across classification and object-detection tasks

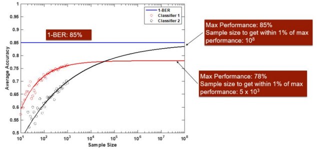
AI/ML lifecycle

Model-agnostic metrics that bound real-world performance:
- relevance/completeness/coverage
- metafeatures (data complexity)



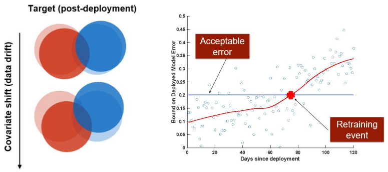
premodel / exploratory

Model-specific metrics that guide model selection and training:
- dataset sufficiency
- data/model complexity mismatch



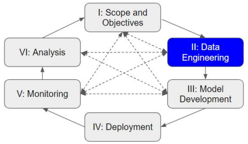
model selection / training

Metrics for post-deployment monitoring of data with bounds on model performance to guide retraining:
- dataset-shift metrics
- model performance bounds under covariate shift
- guidance on sampling to assess model error and model retraining

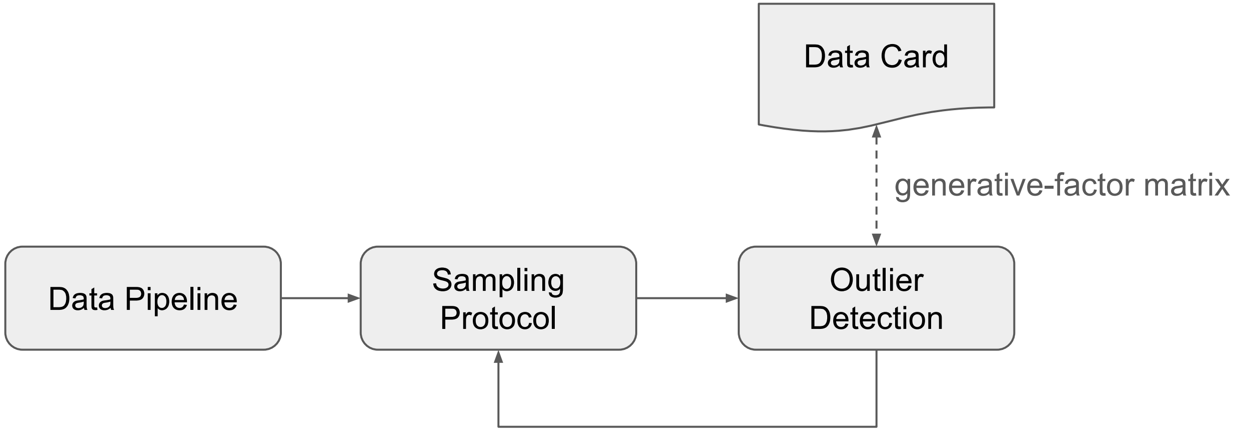


post-deployment / monitoring

Data Engineering

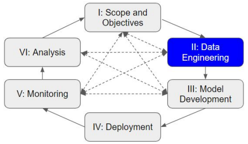


Sampling Protocols: Coverage Assessment



- coverage and completeness for AI/ML assurance can be defined in terms of a matrix of generative factors [cf. Nagy, NAWCWD TP 8864, April 2022 and Hawkins et al., AMLAS v1.1, March 2021] that should be recorded as part of the data card and should drive sampling protocols (e.g., stratified sampling)
- model- and label-independent outlier detection identifies samples that are not near the manifold of sampled data
- when detected outliers represent combinations of generative factors that should be sampled for completeness, the data pipeline or sampling protocol is undersampling that region of the manifold
- the data pipeline and sampling protocol should be refined together with the matrix of generative factors until data required for completeness are no longer characterized as outliers

Data Engineering



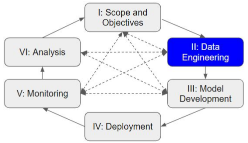
Sampling Protocols: Coverage Assessment

Metrics

Unsupervised Anomaly/Outlier/Out-of-Distribution(OOD) Detection

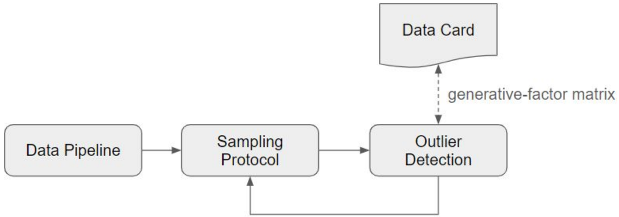
- Auto-Encoder (AE) Reconstruction Error
- Variational Auto-Encoder (VAE) Reconstruction Error
- Auto-Encoding Gaussian Mixture Model (AEGMM) - Zong et al. (2018)
- Variational Auto-Encoding Gaussian Mixture Model (VAEGMM) - Zong et al. (2018)
- Log Likelihood Ratio (LLR) between Generative Models - Ren et al. (2019)

ARiA



Example

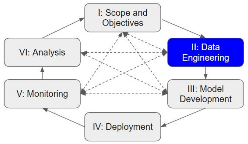
Outlier Detection for Coverage Assessment



- ML system for ground-vehicle detection and classification from aerial LWIR sensor
- matrix of generative factors includes (1) variable backgrounds, (2) variable pixels on target, (3) variable grazing angle, etc.
- outlier detection identifies samples in dataset with grazing angle away from nadir and few pixels on target
- updates to data pipeline and/or sampling protocol required for assurance over desired range of generative factors

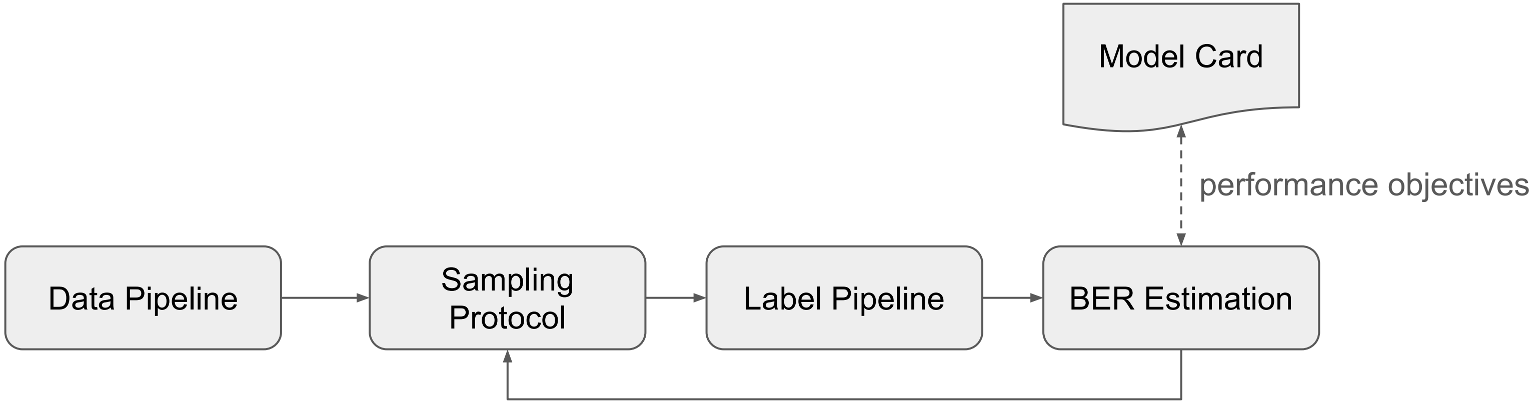
ML model developers will often discard outliers during training to gain robustness and enhance generalization

ARiA



Data Engineering

Exploratory Data Analysis: Data Complexity

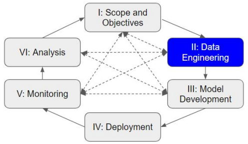


- operational requirements for an ML model should specify performance requirements determined by such factors as safety (e.g., based on a hazard assessment and determination of acceptable risk for the ML model component of a system) as part of the model-card authoring carried out through the ML lifecycle
- the minimum achievable error for a classifier - the Bayes Error Rate (BER) - can be estimated directly from the data
- if the overlap between class distributions is too great, the BER may exceed operational requirements
- when BER exceeds operational requirements, no model can consistently achieve the required performance
- in such cases, the requirements must be refined and/or the task for the ML model reposed - potentially identifying the specific samples that result in distribution overlap

ARiA

Data Engineering

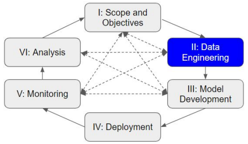
Exploratory Data Analysis: Data Complexity



Metrics

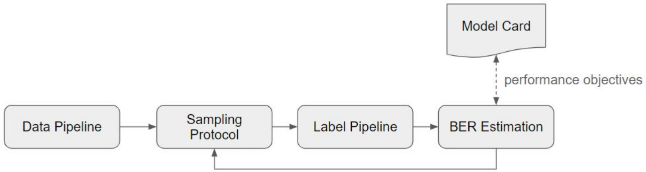
Bounds on BER from empirical estimates of D_p divergence or other *f*-divergences

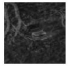
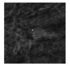
- Nonparametric estimation of D_p divergence from minimum spanning trees - Berisha et al. (2015)

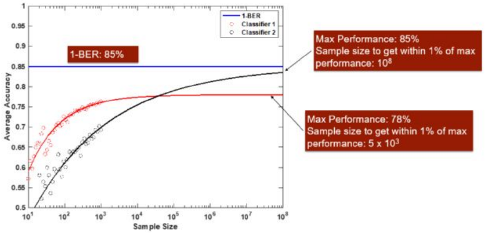


Example

BER estimation to refine operational requirements



BMP2	
SUV	
PICKUP	
VEHICLE	
BRDM2	



- ML system for multiclass ground-vehicle classification from helicopter-based FLIR sensor (SWIR band)
- BER estimate bounds maximum accuracy at 85%, which is below the specified operational requirement for the ML system as determined by functional-hazard assessment of the tactical system
- error can be reduced by combining nontarget overlapping classes - as shown by iterative BER estimation
- refinement of operational requirements in terms of target/nontarget classes enables problem to be reposed in a manner that can be achieved by realizable ML models

Use of model-independent metafeatures in the initial stage of the ML lifecycle can refine objectives early, eliminating wasted effort and decreasing overall development time