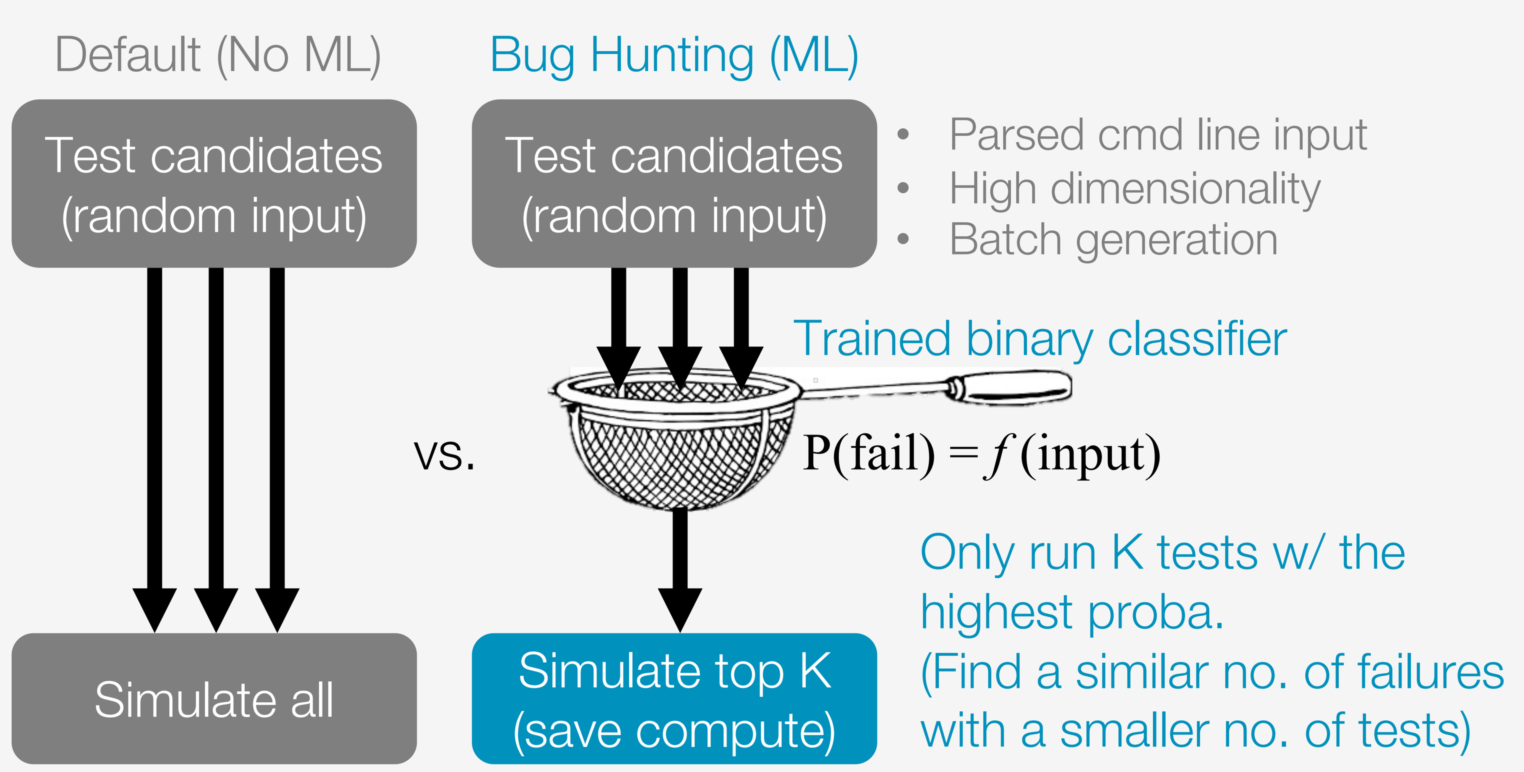
Data-centric ML pipeline for data drift and data preprocessing

Hongsup Shin
hongsup.shin@arm.com
arm

ML enables efficient hardware verification

- The goal of hardware verification is to find almost all design bugs in time to achieve near bug-free design.
- But hardware design space is massive. Thus, engineers use constrained random testing; they generate random test inputs to probe various design spaces. Each test returns pass/failure where a failure means a bug is found.
- We use ML to increase efficiency in hardware verification by guiding the testing behavior.



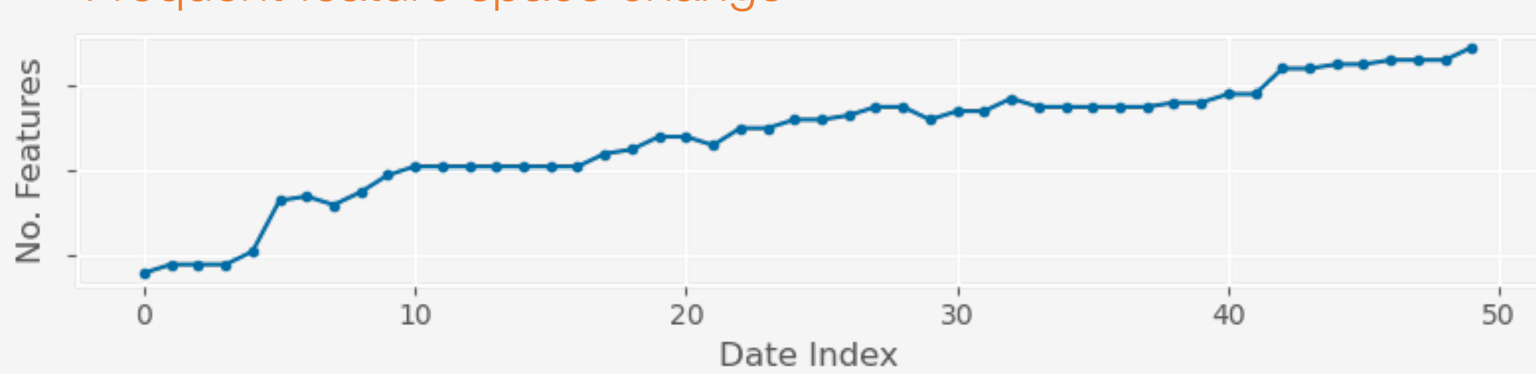
Data problems cause most MLOps issues

Data preprocessing

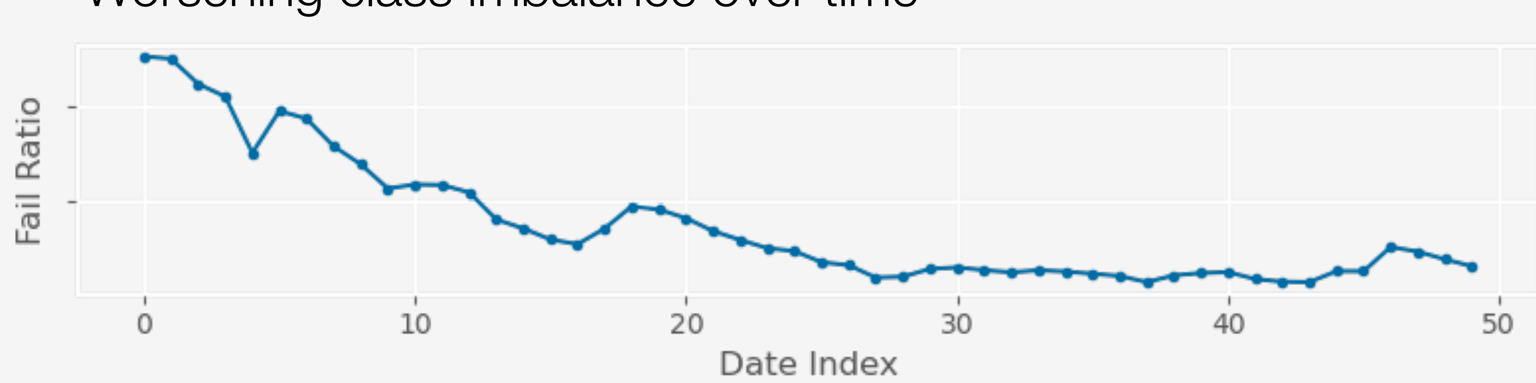
- Need for dtype correction (e.g., “True”)
- Difficult-to-understand features w/ high dimensionality
- Complex regex patterns in string features
- Multiple interpretations available for categorical features

Data drift

- Frequent feature space change



- Worsening class imbalance over time



- dtype of a feature can change over time: (e.g., bool -> str)

Automation feasibility

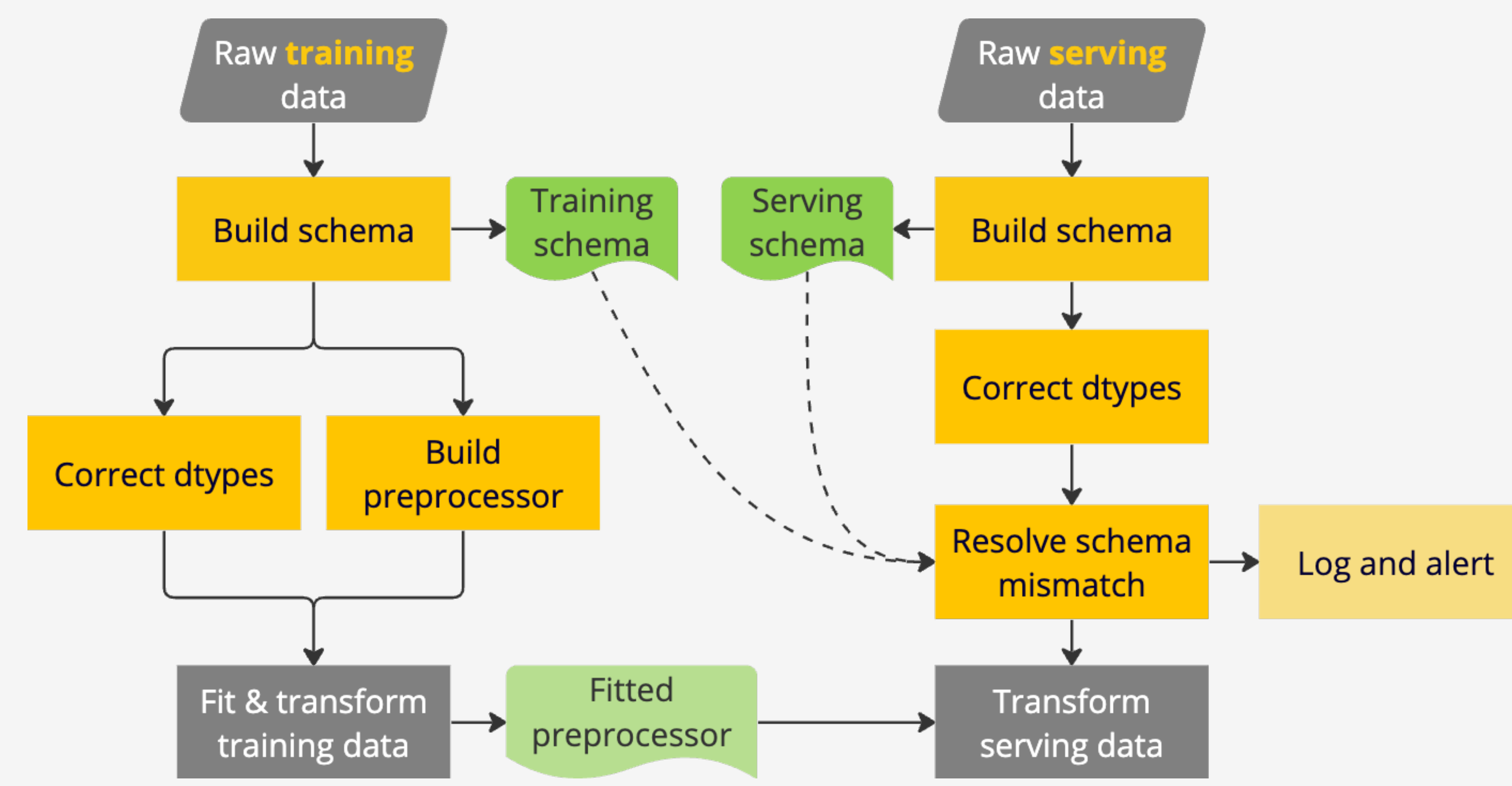
- Frequent interventions from domain experts required to understand any changes in data
- Brittle ML pipeline due to frequent data drift
- Heavy reliance on domain experts: delayed early deployment

Data-centric ML pipeline

Principles

- Data-driven: the ML pipeline preprocesses raw data based on the contents of the data (less dependent on domain experts)
- Flexible & robust: adaptive to changes in training data
- Observable: data preprocessing is transparent and trackable
- Automated: digests raw data automatically

Overview



Schema for monitoring, casting, and preprocessing

	dtypes		
	Inferred	Casting	Preprocessor
Feature 0	“mixed”	dtype('<U')	“lists”
Feature 1	"mixed-integer-U"	dtype('<U')	“nominal_str”
…			

- Generated every time new data arrives (training and serving)
- Infer granular dtypes using pandas.api.types.infer_dtype and numpy.dtype.kind
- Pre-defined mapping for translating the inferred dtypes to others

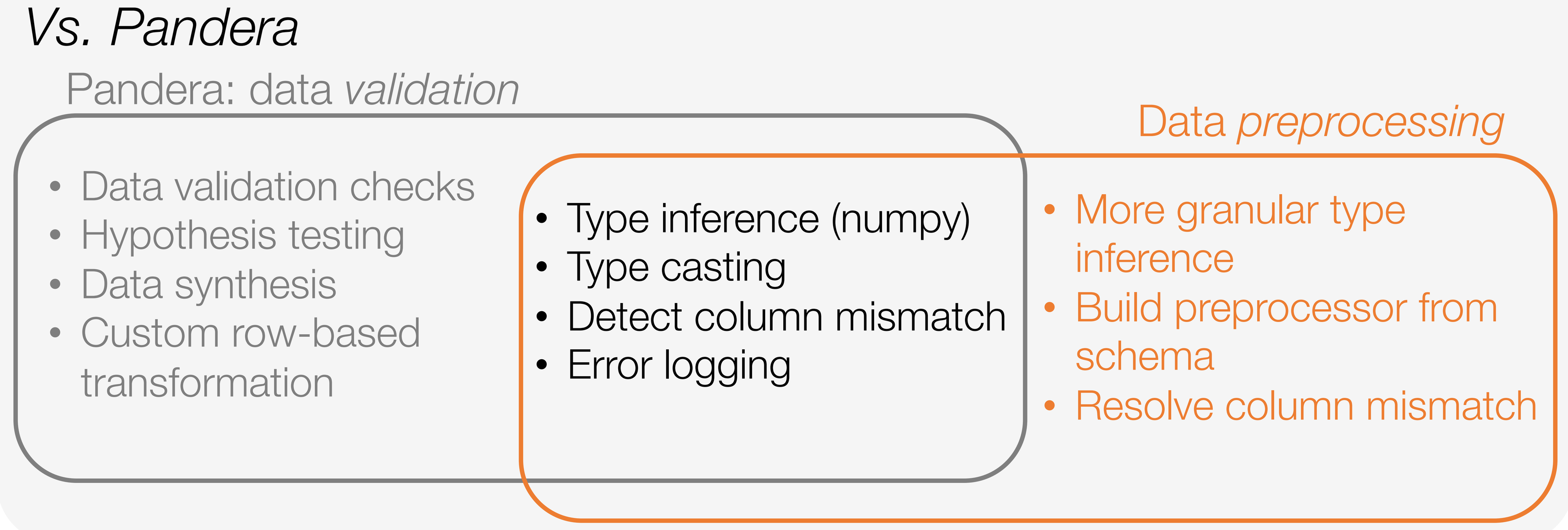
From schema to preprocessor

- Backbone: a scikit-learn pipeline with a ColumnTransformer step
- Pre-defined preprocessing methods (e.g., OneHotEncoder for “nominal_str”) for all dtypes fetched to build column transformer
- Easy to observe and change data preprocessing methods

Data-drift handling

- Not all data drifts are significant
- Schema built during serving and compared with training schema
- First, mismatches are resolved (using dummies) and serving job is run. Any mismatches are logged and trigger alerts.

Vs. Pandera



Data (preprocessing) tuning matters

Why data tuning makes sense

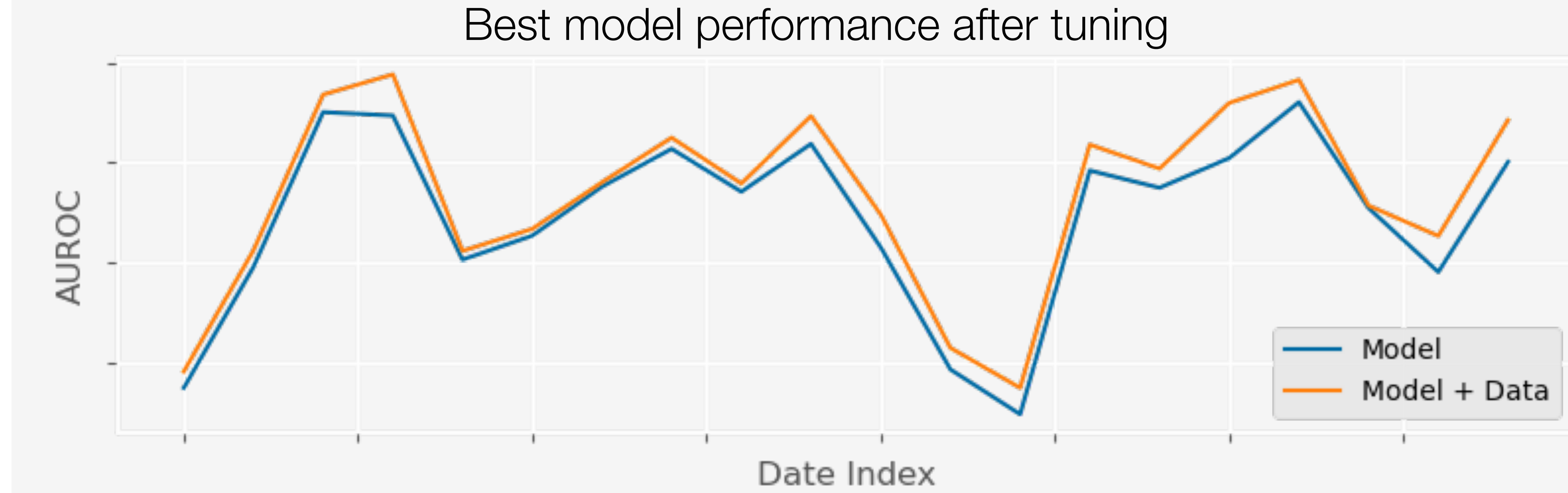
- Data tuning: tuning data preprocessing methods like model hyperparameters
- Difficult-to-understand features (specific to a hw component)
- Multiple interpretations available
- Risks from data-driven schema inference w/o domain knowledge
- Interpretability is not important yet

Benchmark experiments

- 20 post-deployment datasets (~500k rows per dataset) w/ roughly 3:1 time-dependent train-test split
- Lightgbm (sklearn API) w/ daskML randomized search
- 8 different category encoders are used for data tuning (model hyperparameters are kept identical between groups)

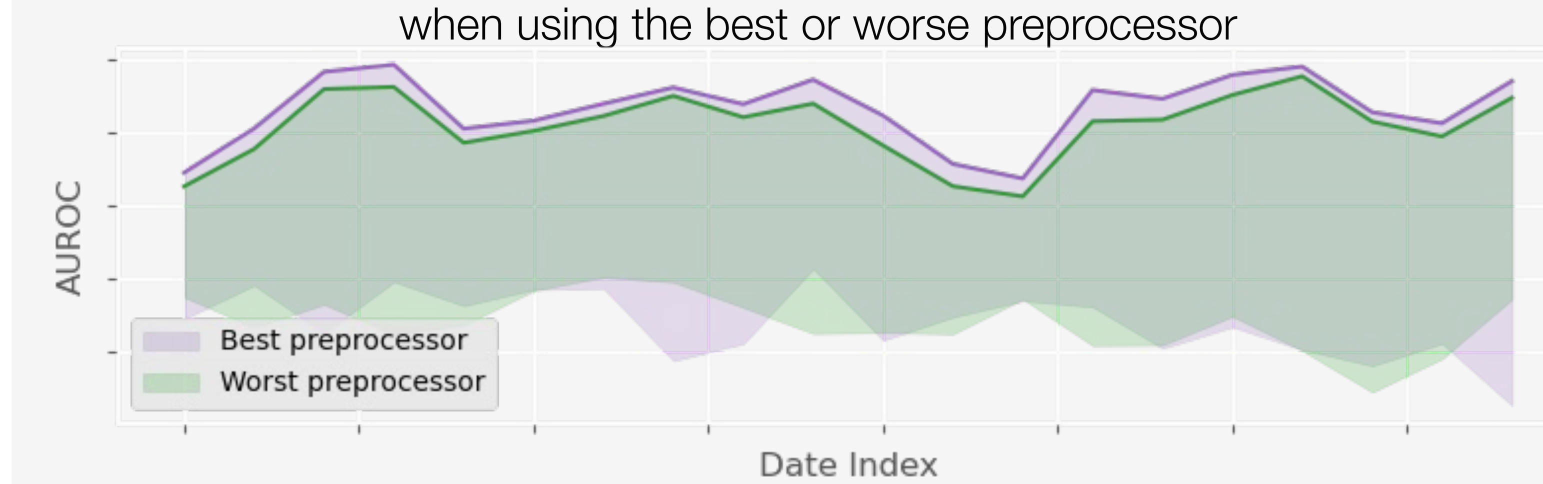
Tuning model & data together > tuning model alone

Best model performance after tuning



How data is preprocessed affects model performance

Min-max performance range from 100 hyperparameter searches when using the best or worse preprocessor



Remaining challenges

Compute and pipeline structure

- Difficult to tune data preprocessing methods without any redundant computation unless transformed data is pre-computed and saved (numerous combos are possible)
- Difficult to configure a streamlined ML pipeline with multiple transformed datasets
- Even more challenging to build a pipeline with cross validation without any data leakage
- Generally challenging to solve interoperability issues when using multiple libraries in a highly customized setting

Interplay between data and model

- Why some features prefer specific preprocessing methods over others?
- What is the role of data preprocessing when tuning model hyperparameters?