SCIENCE
PASSION
TECHNOLOGY

Architecture of ML Systems*
03 Model Selection and Debugging

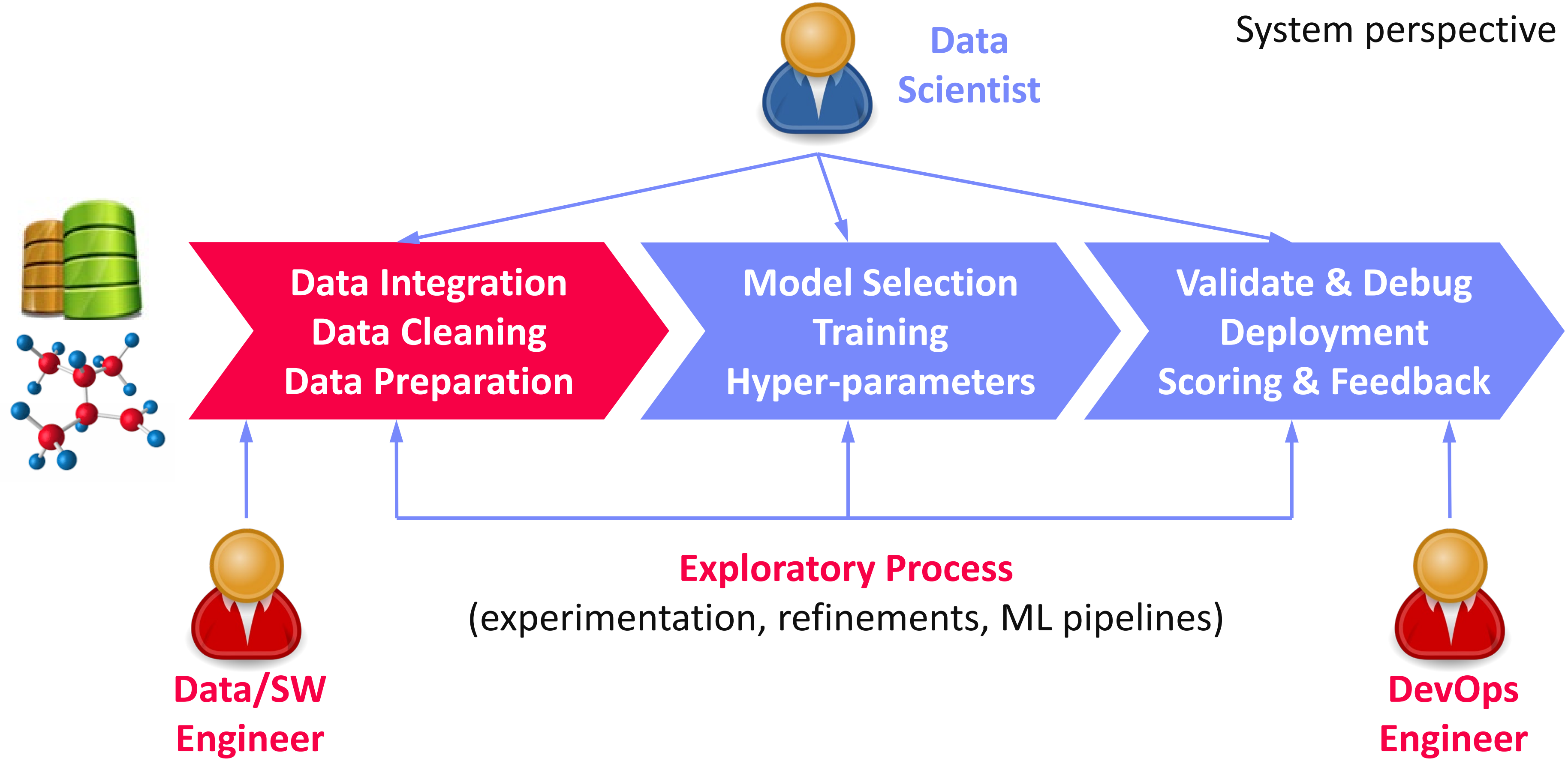
Matthias Boehm

Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMK endowed chair for Data Management

Last update: Aug 25, 2022

Recap: The Data Science Lifecycle

Data-centric View:
Application perspective
Workload perspective
System perspective



Data Scientist

Data Integration
Data Cleaning
Data Preparation

Model Selection
Training
Hyper-parameters

Validate & Debug
Deployment
Scoring & Feedback

Exploratory Process
(experimentation, refinements, ML pipelines)

Data/SW Engineer

DevOps Engineer

Agenda

- **Model Selection Techniques**
- **Model Management & Provenance**
- **Model Debugging and Explainability**
- **Model Bias & Fairness Constraints**

Model Selection Techniques

AutoML Overview

[Chris Thornton, Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. **KDD 2013**]

- **#1 Model Selection**
  - Given a dataset and ML task
    (e.g., classification or regression)
  - Select the model (type) that performs best
    (e.g.: LogReg, Naïve Bayes, SVM, Decision Tree, Random Forest, DNN)

$$A^* \in \operatorname*{argmin}_{A \in \mathcal{A}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A, \mathcal{D}^{(i)}_{\mathrm{train}}, \mathcal{D}^{(i)}_{\mathrm{valid}}),$$

- **#2 Hyper Parameter Tuning**
  - Given a model and dataset,
    find best hyper parameter values
    (e.g., learning rate, regularization, kernels, kernel parameters, tree params)

$$A^*{}_{\boldsymbol{\lambda}^*} \in \operatorname*{argmin}_{A^{(j)} \in \mathcal{A}, \boldsymbol{\lambda} \in \boldsymbol{\Lambda}^{(j)}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A^{(j)}_{\boldsymbol{\lambda}}, \mathcal{D}^{(i)}_{\mathrm{train}}, \mathcal{D}^{(i)}_{\mathrm{valid}}).$$

- **Validation: Generalization Error**
  - Goodness of fit to held-out data (e.g., 80-20 train/test)
  - Cross validation (e.g., leave one out → k=5 runs w/ 80-20 train/test)

→ AutoML Systems/Services

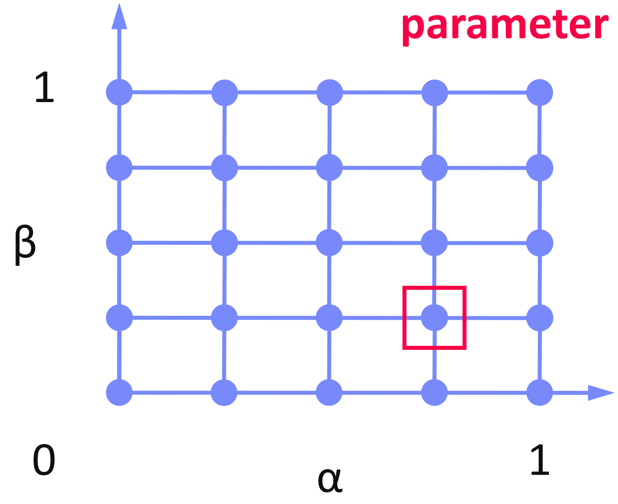
- Often providing both model selection and hyper parameter search
- Integrated ML system, often in distributed/cloud environments

Basic Grid Search

`gridSearch()` `GridSearchCV()`

- **Basic Approach**
  - Given n hyper parameters λ1, …, λn with domains Λ1, …, Λn
  - Enumerate and evaluate parameter space $\Lambda \subseteq \Lambda_1 \times \ \dots \ \times \Lambda_n$ (often strict subset due to dependency structure of parameters)
  - Continuous hyper parameters → discretization
    - Equi-width
    - Exponential (e.g., regularization 0.1, 0.01, 0.001, etc)
  - **Problem:** Only applicable with small domains
- **Heuristic: Monte-Carlo (random search, anytime)**

Non-convex or unknown parameter space

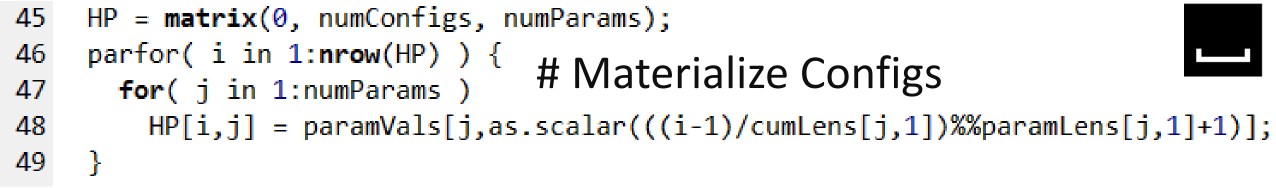


Basic Grid Search, cont.

- **Example Adult Dataset** (train 32,561 x 14)
  - Binary classification (>50K), https://archive.ics.uci.edu/ml/datasets/adult
  - **#1** MLogReg defaults w/ one-hot categoricals **Accuracy (%): 82.35**
  - **#2** MLogReg defaults w/ one-hot + binning **Accuracy (%): 84.73**
  - **#3** GridSearch MLogReg: **Accuracy (%): 90.07**

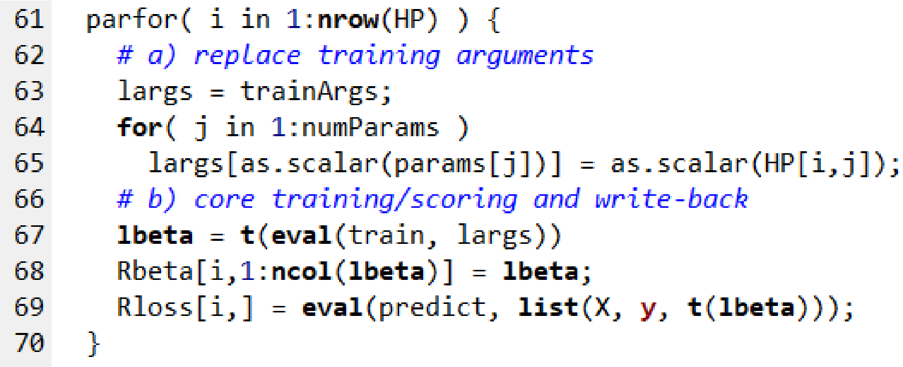
```
params = list("icpt", "reg", "numBins");
paramRanges = list(seq(0,2), 10^seq(3,-6), 10^seq(1,4));
```

- **Example SystemDS** `gridSearch`



```
HP = matrix(0, numConfigs, numParams);
parfor( i in 1:nrow(HP) ) {
  for( j in 1:numParams )
    HP[i,j] = paramVals[j,as.scalar(((i-1)/cumLens[j,1])%%paramLens[j,1]+1)];
}
```

# Materialize Configs



```
parfor( i in 1:nrow(HP) ) {
  # a) replace training arguments
  largs = trainArgs;
  for( j in 1:numParams )
    largs[as.scalar(params[j])] = as.scalar(HP[i,j]);
  # b) core training/scoring and write-back
  lbeta = t(eval(train, largs))
  Rbeta[i,1:ncol(lbeta)] = lbeta;
  Rloss[i,] = eval(predict, list(X, y, t(lbeta)));
}
```

05 Data- and Task-Parallel Execution

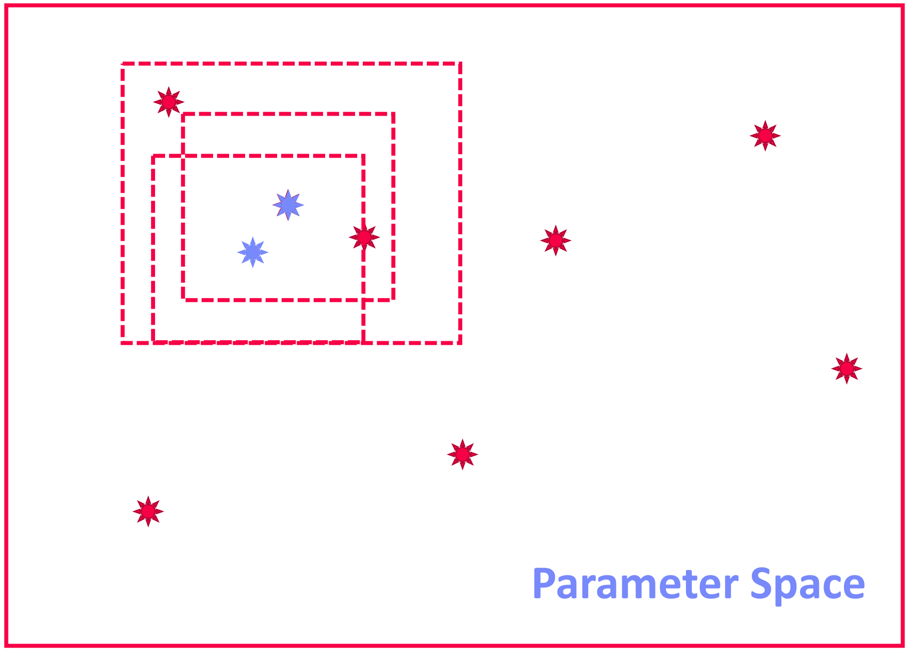
Basic Iterative Algorithms

- **Simulated Annealing**
  - Decaying temperature schedules: **$T_{k+1} = \alpha \cdot T_k$**
  - **#1** Generate neighbor in **ε-env** of old point
  - **#2** Accept better points and worse points w/

$$P(T_k) = \frac{1}{1 + \exp((f' - f)/T_k)}$$

Exploration vs exploitation

- **Recursive Random Search**
  - Repeated restart
  - Sample and evaluate points
  - Determine best and shrink area if optimum unchanged
  - Realign area if new optimum found

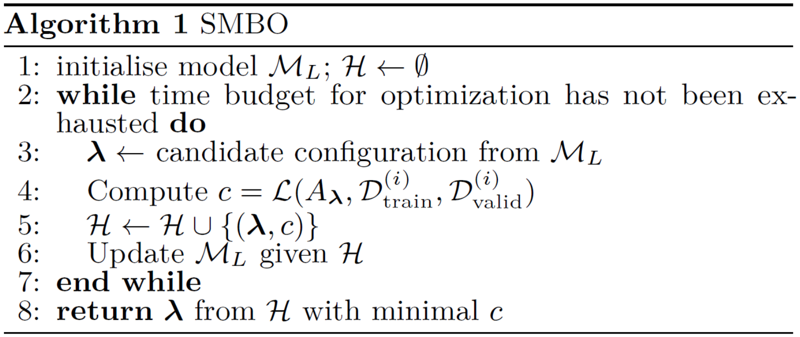


[Tao Ye, Shivkumar Kalyanaraman: A recursive random search algorithm for large-scale network parameter configuration. **SIGMETRICS 2003**]

Bayesian Optimization

[Chris Thornton, Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. **KDD 2013**]

- **Overview BO**
  - Sequential Model-Based Optimization
  - Fit a probabilistic model based on the first n-1 evaluated hyper parameters
  - Use model to select next candidate
  - **Gaussian process (GP)** models, or tree-based Bayesian Optimization



Algorithm 1 SMBO

```
initialise model $\mathcal{M}_L$; $\mathcal{H} \leftarrow \emptyset$
while time budget for optimization has not been exhausted do
    $\boldsymbol{\lambda} \leftarrow$ candidate configuration from $\mathcal{M}_L$
    Compute $c = \mathcal{L}(A_{\boldsymbol{\lambda}}, \mathcal{D}^{(i)}_{\text{train}}, \mathcal{D}^{(i)}_{\text{valid}})$
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{\lambda}, c)\}$
    Update $\mathcal{M}_L$ given $\mathcal{H}$
end while
return $\boldsymbol{\lambda}$ from $\mathcal{H}$ with minimal $c$
```

- **Underlying Foundations**
  - The posterior probability of a model M given evidence E is proportional to the likelihood of E given M multiplied by prior probability of M
  - **Prior knowledge:** e.g., smoothness, noise-free
  - **Maximize acquisition function:**
    GP high objective (exploitation) and high prediction uncertainty (exploration)

$$P(M|E) = P(E|M)P(M)/P(E)$$

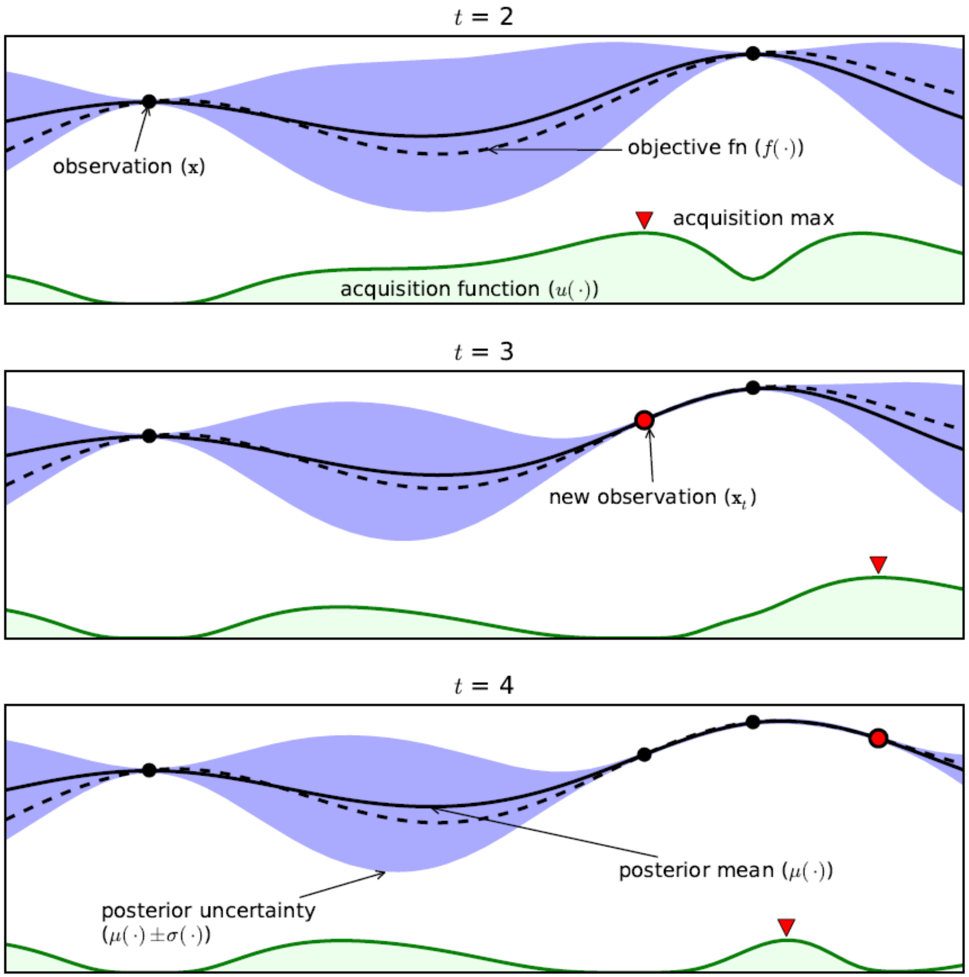
$$\rightarrow$$

$$\underbrace{P(M|E)}_{\text{after}} \propto \underbrace{P(E|M)}_{\text{next}}\underbrace{P(M)}_{\text{before}}$$

Bayesian Optimization, cont

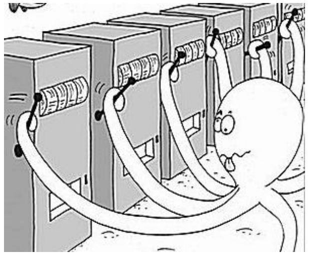
- **Example 1D Problem**
  - Gaussian Process
  - 4 iterations

[Eric Brochu, Vlad M. Cora, Nando de Freitas: A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. **CoRR 2010**]



Multi-armed Bandits and Hyperband

[**Credit:** blogs.mathworks.com]



- **Overview Multi-armed Bandits**
  - Motivation: model types have different quality
  - Select among k model types → **k-armed bandit problem**
  - Running score for each arm → **scheduling policy**

[Sébastien Bubeck, Nicolò Cesa-Bianchi: Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. **Foundations and Trends in Machine Learning 2012**]

- **Hyperband**
  - Non-stochastic setting, without parametric assumptions
  - Pure exploration algorithm for **infinite-armed bandits**
  - Based on **Successive Halving**
    - Successively discarding the worst-performing half of arms
    - Extended by doubling budget of arms in each iteration (no need to configure k, random search included)

[Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, Ameet Talwalkar: Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. **JMLR 2017**]

Selected AutoML Systems

- **Auto Weka**
  - Bayesian optimization with 28 learners, 11 ensemble/meta methods
- **Auto Sklearn**
  - Bayesian optimization with 15 classifiers, 14 feature prep, 4 data prep
- **TuPaQ**
  - **Multi-armed bandit** and large-scale
- **TPOT**
  - Genetic programming
- **Other Services**
  - Azure ML, Amazon ML
  - Google AutoML, H20 AutoML

[Chris Thornton et al: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. **KDD 2013**]

[Lars Kotthoffet al: Auto-WEKA 2.0: Automatic model selection and hyper-parameter optimization in WEKA. **JMLR 2017**]

[Matthias Feurer et al: Auto-sklearn: Efficient and Robust Automated Machine Learning. **Automated Machine Learning 2019**]

[Evan R. Sparks, Ameet Talwalkar, Daniel Haas, Michael J. Franklin, Michael I. Jordan, Tim Kraska: Automating model search for large scale machine learning. **SoCC 2015**]

[Randal S. Olson, Jason H. Moore: TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning. **Automated Machine Learning 2019**]

[Hantian Zhang, Luyuan Zeng, Wentao Wu, Ce Zhang: How Good Are Machine Learning Clouds for Binary Classification with Good Features? **CoRR 2017**]

Selected AutoML Systems, cont.

- **Alpine Meadow**
  - Logical and physical ML pipelines
  - **Multi-armed bandit** for pipeline selection
  - **Bayesian optimization** for hyper-parameters

[Zeyuan Shang et al: Democratizing Data Science through Interactive Curation of ML Pipelines. **SIGMOD 2019**]

- **Dabl** (Data Analysis Baseline Library)
  - Tools for simple data preparation and ML training
  - **Hyperband** (successive halving) for optimization

[https://amueller.github.io/dabl/dev/user_guide.html]

- **BOHB**
  - **Bayesian optimization** & **hyperband**
  - Queue-based **parallelization** of successive halving

[Stefan Falkner, Aaron Klein, Frank Hutter: BOHB: Robust and Efficient Hyper-parameter Optimization at Scale. **ICML 2018**]

- **AutoML** (https://www.automl.org/)
  Paper Collections/Benchmarks
  - HPOBench/NASBench

AutoML Freiburg-Hannover

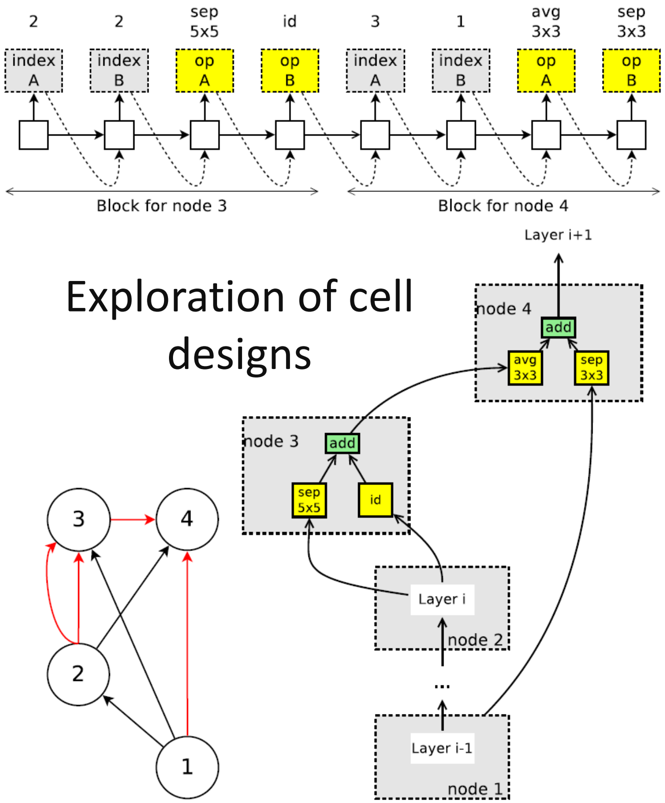
Neural Architecture Search

- **Motivation**
  - Design neural networks (type of layers / network) is often trial & error process
  - Accuracy vs necessary computation characterizes an architecture

  ➔ **Automatic neural architecture search**

- **#1 Search Space of Building Blocks**
  - Define possible operations (e.g., identity, 3x3/5x5 separable convolution, avg/max pooling)
  - Define approach for connecting operations (pick 2 inputs, apply op, and add results)

[Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, Jeff Dean: Efficient Neural Architecture Search via Parameter Sharing. **ICML 2018**]



Exploration of cell designs

Neural Architecture Search, cont.

- **#2 Search Strategy**
  - Classical evolutionary algorithms
  - Recurrent neural networks (e.g., LSTM)
  - Bayesian optimization (with special distance metric)

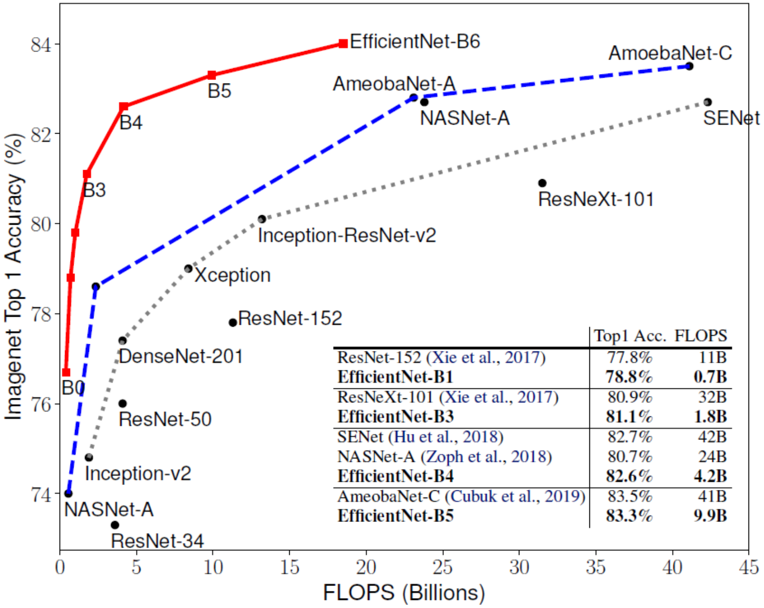
[Barret Zoph, Quoc V. Le: Neural Architecture Search with Reinforcement Learning. **ICLR 2017**]

[Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabás Póczos, Eric P. Xing: Neural Architecture Search with Bayesian Optimisation and Optimal Transport. **NeurIPS 2018**]

- **#3 Optimization Objective**
  - Max accuracy (min error)
  - Multi-objective (accuracy and runtime)

- **Excursus: Model Scaling**
  - Automatically scale-up small model for better accuracy
  - EfficientNet

[Mingxing Tan, Quoc V. Le: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. **ICML 2019**]



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**78.8%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.1%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.6%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.3%**	**9.9B**

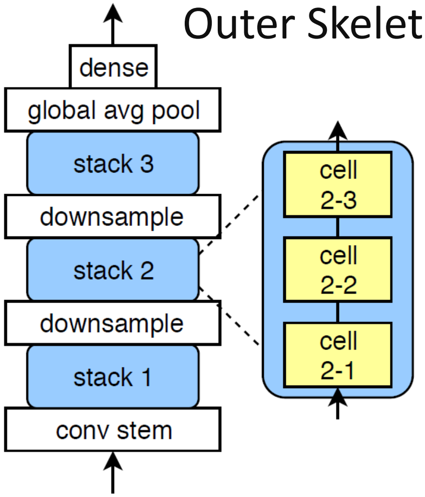
Neural Architecture Search, cont.

- **Problem: Computational Resources**
  - Huge computational requirements for NAS (even on small datasets)

  ➔ **#1 Difficult to reproduce**, and **#2 barrier-to-entry**

- **Excursus: NAS-Bench-101**
  - **423K** unique convolutional architectures
  - Training and evaluated **ALL** architectures, **multiple times** on **CIFAR-10**
  - Shared dataset: **5M trained models**

[Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, Frank Hutter: NAS-Bench-101: Towards Reproducible Neural Architecture Search. **ICML 2019**]

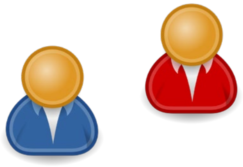


Outer Skeleton

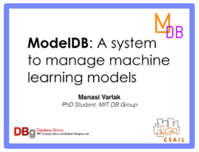
Model Management & Provenance

Overview Model Management

How did you create that model?
Did you consider X?



- **Motivation**
  - Exploratory data science process → trial and error (preparation, feature engineering, model selection)
  - Different personas (data engineer, ML expert, devops)
- **Problems**
  - No record of experiments, insights lost along the way
  - Difficult to reproduce results
  - Cannot search for or query models
  - Difficult to collaborate
- **Overview**
  - Experiment tracking and visualization
  - Coarse-grained ML pipeline provenance and versioning
  - Fine-grained data provenance (data-/ops-oriented)



ModelDB: A system to manage machine learning models

Manasi Vartak
PhD Student, MIT DB Group

DBg Database Group MIT Computer Science and Artificial Intelligence Lab

CSAIL

[Manasi Vartak: ModelDB: A system to manage machine learning models, **Spark Summit 2017**]

Model Management Systems (MLOps)

- **ModelHub**
  - Versioning system for DNN models, including provenance tracking
  - DSL for model exploration and enumeration queries (model selection + hyper parameters)
  - Model versions stored as deltas

[Hui Miao, Ang Li, Larry S. Davis, Amol Deshpande: ModelHub: Deep Learning Lifecycle Management. **ICDE 2017**]

- **ModelDB → Verta.ai**
  - Model and provenance logging for ML pipelines via programmatic APIs
  - Support for different ML systems (e.g., spark.ml, scikit-learn, others)
  - GUIs for capturing meta data and metric visualization

[Manasi Vartak, Samuel Madden: MODELDB: Opportunities and Challenges in Managing Machine Learning Models. **IEEE Data Eng. Bull. 2018**]

[Verta Enterprise MLOps Platform https://www.verta.ai/platform/]

Model Management Systems (MLOps), cont.

- **MLflow**
  - An open source platform for the machine learning lifecycle
  - Use of existing ML systems and various language bindings
  - **MLflow Tracking:** logging and querying experiments
  - **MLflow Projects:** packaging/reproduction of ML pipeline results
  - **MLflow Models:** deployment of models in various services/tools
  - **MLflow Model Registry:** cataloging models and managing deployment

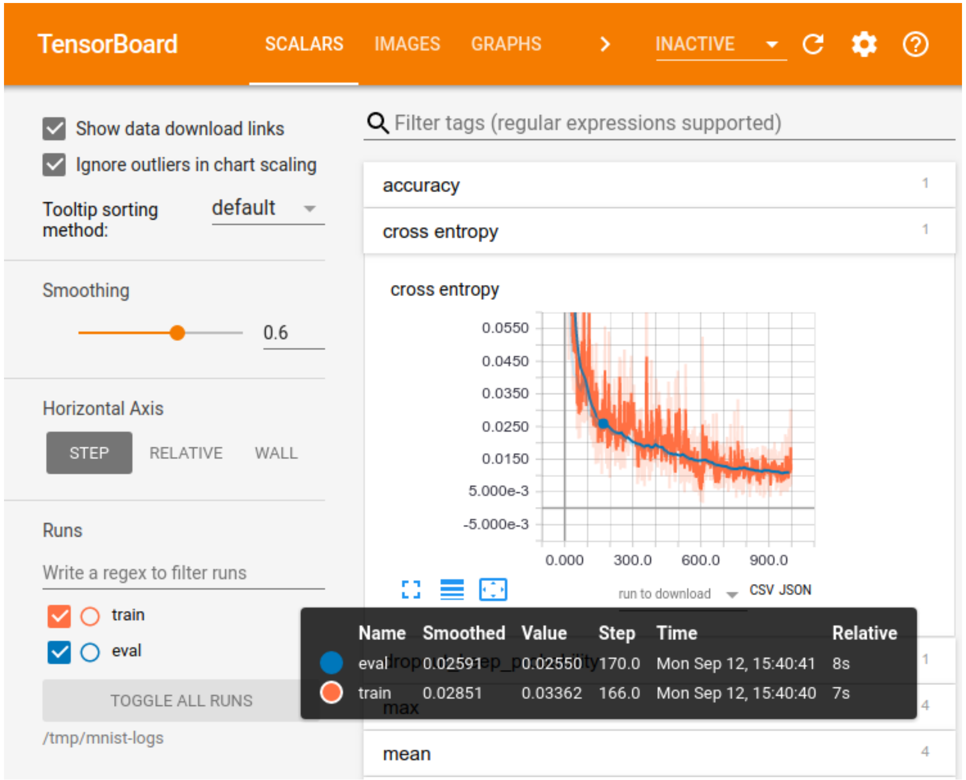
[Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, Corey Zumar: Accelerating the Machine Learning Lifecycle with MLflow. **IEEE Data Eng. Bull. 41(4) 2018**]

[Andrew Chen, Andy Chow, Aaron Davidson, Arjun DCunha, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Clemens Mewald, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Avesh Singh, Fen Xie, Matei Zaharia, Richard Zang, Juntai Zheng, Corey Zumar: Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle. **DEEM@SIGMOD 2020**]

Experiment Tracking

- **TensorFlow: TensorBoard**
  - Suite of visualization tools
  - Explicitly track and write summary statistics
  - Visualize behavior over time and across experiments
  - Different folders for model versioning?

- **Other Tools:**
  - Integration w/ TensorBoard
  - Lots of custom logging and plotting tools



☑ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing: 0.6

Horizontal Axis: STEP

Runs

Write a regex to filter runs

☑ ○ train

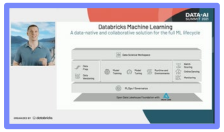
☑ ○ eval

/tmp/mnist-logs

	Name	Smoothed	Value	Step	Time	Relative
●	eval	0.02591	0.02550	170.0	Mon Sep 12, 15:40:41	8s
●	train	0.02851	0.03362	166.0	Mon Sep 12, 15:40:40	7s

[**Credit:** https://www.tensorflow.org/guide/summaries_and_tensorboard]

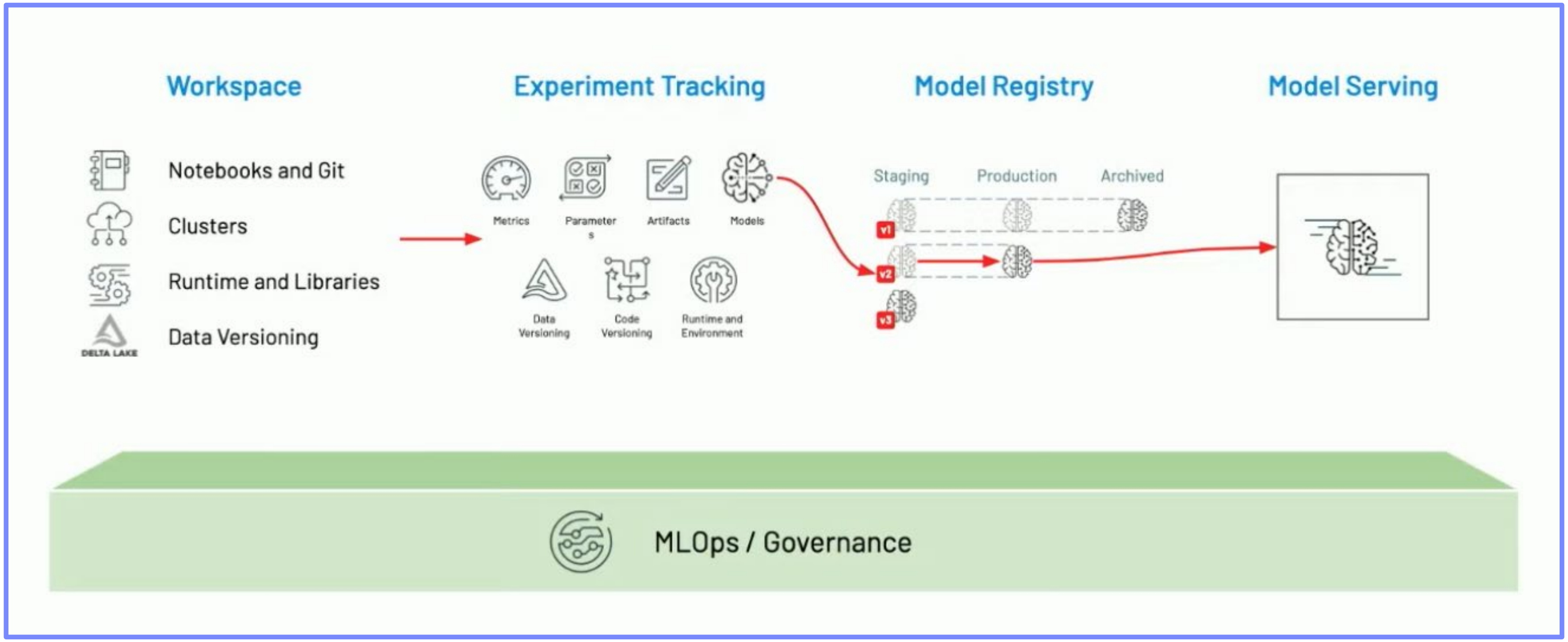
ML Lifecycle Management



- **Databricks Machine Learning**
  - MLOps, Feature Store, AutoML

[Clemens Mewald: Announcing Databricks Machine Learning, Feature Store, AutoML, **Keynote Data+ AI Summit 2021**]

```
MLOps = DataOps + DevOps + ModelOps
```



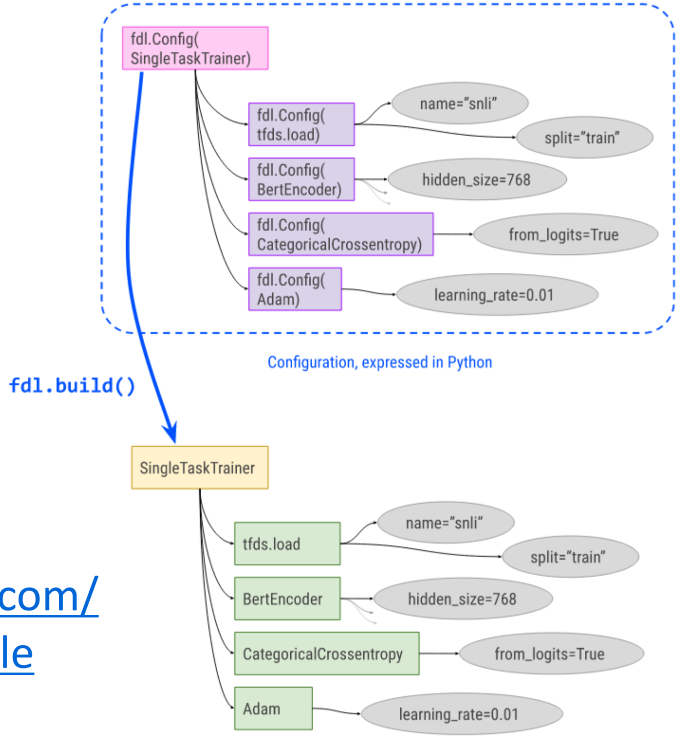
Configuration Management

- **#1 ML Collections**
  - Dictionary-like data structures for configurations of experiments and models (hyper-parameters, loss, optimizer)
  - `ConfigDict` and `FrozenConfigDict`

https://github.com/google/ml_collections

- **#2 Fiddle**
  - **Configurations for model training** with `build()` for creating training instances
  - **Auto-config** for creating a config object from a (control-flow-free) function
  - Explain and visualization

https://github.com/google/fiddle



Provenance for ML Pipelines (fine-grained)

- **DEX: Dataset Versioning**
  - **Versioning of datasets, stored with delta encoding**
  - Checkout, intersection, union queries over deltas
  - Query optimization for finding efficient plans

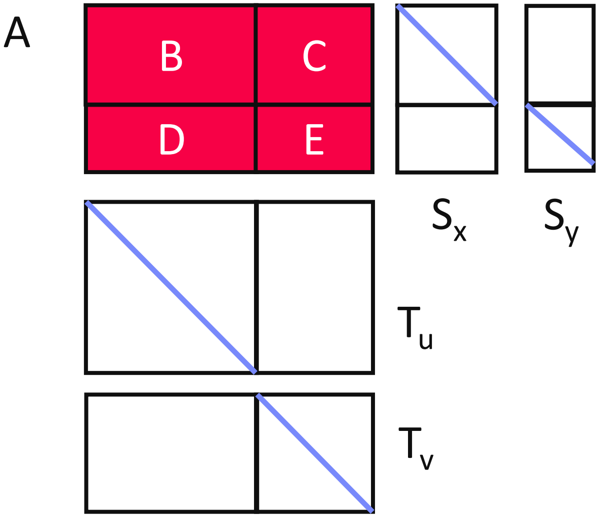
[Amit Chavan, Amol Deshpande: DEX: Query Execution in a Delta-based Storage System. **SIGMOD 2017**]

- **MISTIQUE: Intermediates of ML Pipelines**
  - Capturing, storage, querying of intermediates
  - **Lossy deduplication and compression**
  - Adaptive querying/materialization for finding efficient plans

[Manasi Vartak et al: MISTIQUE: A System to Store and Query Model Intermediates for Model Diagnosis. **SIGMOD 2018**]

- **Linear Algebra Provenance**
  - Provenance propagation by decomposition
  - Annotate parts w/ provenance polynomials (identifiers of contributing inputs + impact)

$$A = S_x B T_u + S_x C T_v + S_y D T_u + S_y E T_v$$



[Zhepeng Yan, Val Tannen, Zachary G. Ives: Fine-grained Provenance for Linear Algebra Operators. **TaPP 2016**]

Provenance for ML Pipelines (coarse-grained)

- **MLflow**
  - Programmatic API for tracking parameters, experiments, and results
  - **autolog()** for specific params

[**Credit:** https://databricks.com/blog/2018/06/05]

```
import mlflow
mlflow.log_param("num_dimensions", 8)
mlflow.log_param("regularization", 0.1)
mlflow.log_metric("accuracy", 0.1)
mlflow.log_artifact("roc.png")
```

- **Flor (on Ground)**
  - DSL embedded in python for managing the workflow development phase of the ML lifecycle
  - DAGs of actions, artifacts, and literals
  - Data context generated by activities in Ground

[Credit: https://rise.cs.berkeley.edu/projects/jarvis/]

[Joseph M. Hellerstein et al: Ground: A Data Context Service. **CIDR 2017**]

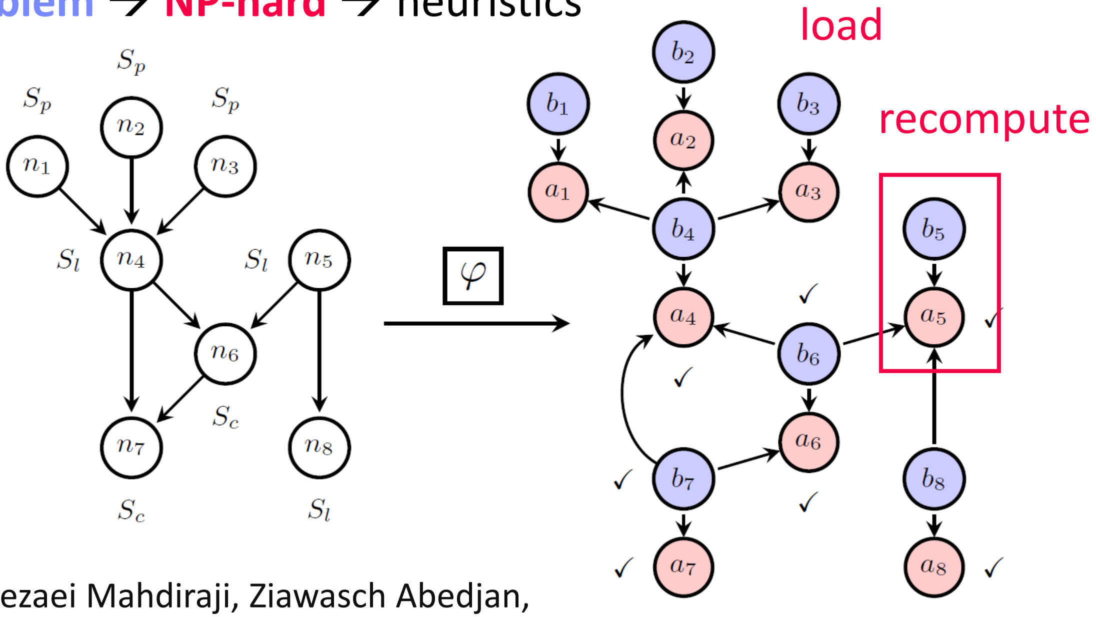
- **Dataset Relationship Management**
  - **Reuse**, **reveal**, **revise**, **retarget**, **reward**
  - Code-to-data relationships (data provenance)
  - Data-to-code relationships (potential transforms)

[Zachary G. Ives, Yi Zhang, Soonbo Han, Nan Zheng,: Dataset Relationship Management. **CIDR 2019**]

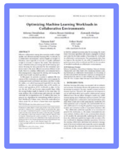
Provenance for ML Pipelines (coarse-grained), cont.

- **HELIX**
  - Goal: focus on iterative development w/ small modifications (trial & error)
  - Caching, reuse, and recomputation
  - Reuse as Max-Flow problem → NP-hard → heuristics
  - Materialization to disk for future reuse

[Doris Xin, Stephen Macke, Litian Ma, Jialin Liu, Shuchen Song, Aditya G. Parameswaran: Helix: Holistic Optimization for Accelerating Iterative Machine Learning. **PVLDB 2018**]



- **Collaborative Optimizer**



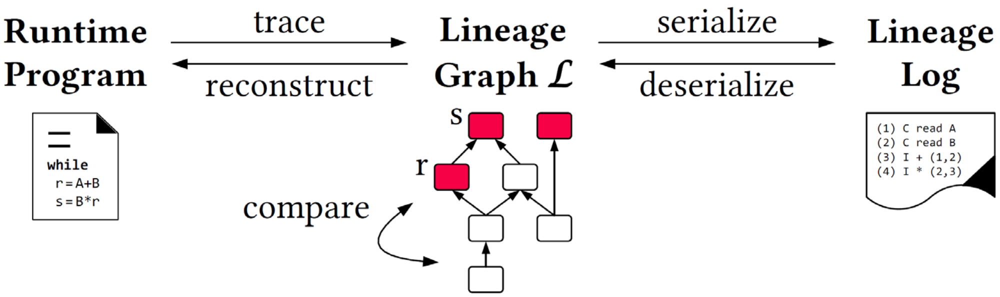
[Behrouz Derakhshan, Alireza Rezaei Mahdiraji, Ziawasch Abedjan, Tilmann Rabl, Volker Markl: Optimizing Machine Learning Workloads in Collaborative Environments. **SIGMOD 2020**]

Lineage Tracing & Reuse in SystemDS

- **Problem**
  - **Exploratory data science** (data preprocessing, model configurations)
  - **Reproducibility** and **explainability** of trained models (data, parameters, prep)

  ➔ **Lineage/Provenance as Key Enabling Technique:**
  Model versioning, reuse of intermediates, incremental maintenance, auto differentiation, and debugging (query processing over lineage)

- **Efficient Lineage Tracing**
  - Tracing of inputs, literals, and **non-determinism**
  - **Trace lineage of logical operations**
  - **Deduplication** for loops/functions
  - Program/output reconstruction
- **Lineage-Based Reuse**



```
while
  r=A+B
  s=B*r
```

```
(1) C read A
(2) C read B
(3) I + (1,2)
(4) I * (2,3)
```

[Arnab Phani, Benjamin Rath, Matthias Boehm: LIMA: Fine-grained Lineage Tracing and Reuse in Machine Learning Systems, **SIGMOD 2021**]

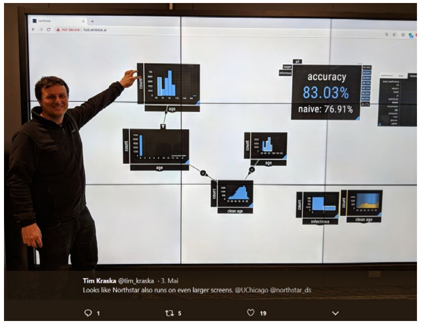
Model Debugging and Explainability

Similar to Software Testing

Focus on Benchmarks, Assessment, Monitoring, Model Improvements, Model Understanding

Overview Model Debugging

[**Credit:** twitter.com/tim_kraska]

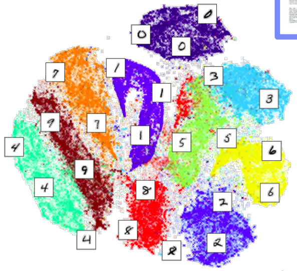


Tim Kraska @tim_kraska · 3. Mai
Looks like Northstar also runs on even larger screens. @UChicago @northstar_ds

1 5 19

[Andrew Crotty et al: Vizdom: Interactive Analytics through Pen and Touch. **PVLDB 2015**]

- **#1 Understanding via Visualization**
  - Plotting of predictions / interactions
  - Combination with dimensionality reduction into 2D:
    - **Autoencoder**
    - **PCA** (principal component analysis)
    - **t-SNE** (T-distributed Stochastic Neighbor Embedding)
  - Input, intermediate, and output layers of DNNs



[**Credit:** nlml.github.io/in-raw-numpy/in-raw-numpy-t-sne/]

- **#2 Validation, Explainability, Fairness via Constraints**
  - Establish assertions and thresholds for automatic validation and alerts w.r.t. **accuracy**, **bias**, and other metrics
  - Generate succinct representations (e.g., rules) as **explanation**
  - Impose constraints like monotonicity for ensuring **fairness**

Basic Model-Specific Statistics

- **Regression Statistics**
  - Average response and stddev, average residuals stddev residuals
  - R2 (coeff of determination) with and without bias, etc
- **Classification Statistics**
  - Classical: recall, precision, F1-score
  - Visual: confusion matrix (correct vs predicated classes) ➔ understand performance wrt individual classes / bins of continuous vars
  - Example Mnist
  - Mispredictions might also be visualized via dimensionality reduction

predicted label

correct label

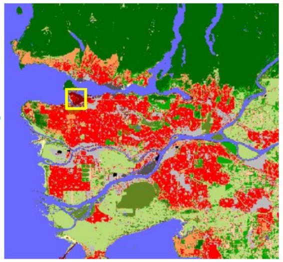
	0	1	2	3	4	5	6	7	8	9
0	21									
1		25								
2			15							
3				76						
4					23					12
5						36				
6							24			
7								31		37
8									42	
9					8			11		53

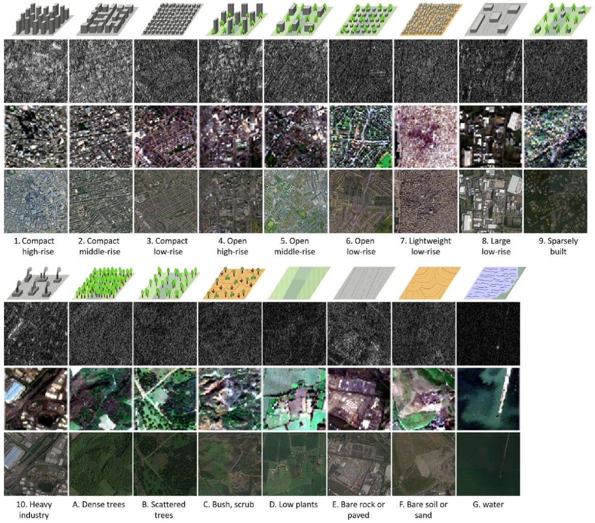
Excursus: DLR Earth Observation Use Case

[Xiao Xiang Zhu et al: So2Sat LCZ42: A Benchmark Dataset for the Classification of Global Local Climate Zones. **GRSM 2020**]

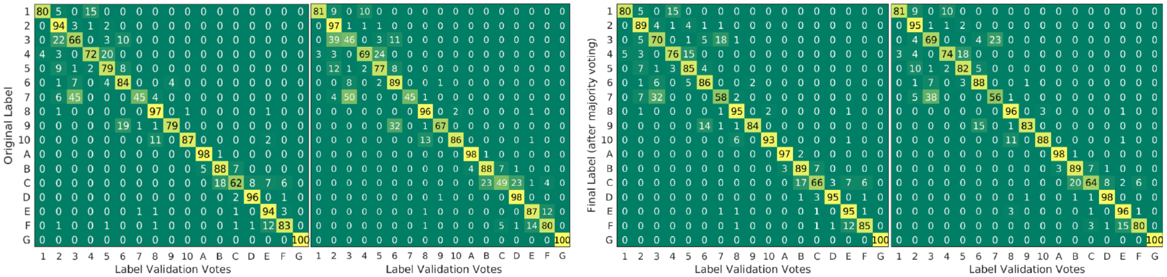
[So2Sat LC42 Dataset https://mediatum.ub.tum.de/1454690]

- **Data and ML Pipelines**
  - ESA Sentinel-1/2 datasets → 4PB/year
  - Training of local climate zone classifiers on So2Sat LCZ42 (15 experts, 400K instances, 10 labels each, 85% confidence, ~55GB H5)
  - ML pipeline: preprocessing, ResNet18, climate models





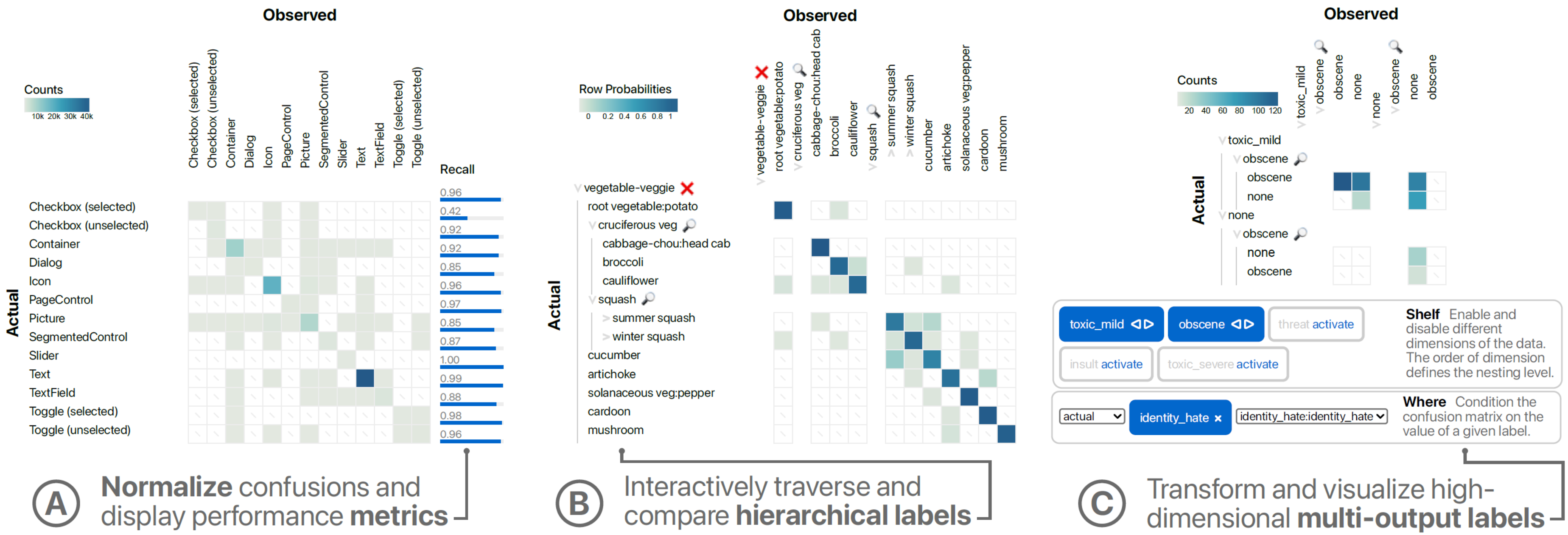
- **Label Creation/ Validation**
  - Team learning
  - Labeling w/ checks
  - Label validation
  - Quantitative validation w/ 10 expert votes on correctness



Confusion Matrices, cont.

- **Generalized Confusion Matrices**
  - Hierarchical, Multi-label Data

[Jochen Görtler et al: **Neo:** Generalizing Confusion Matrix Visualization to Hierarchical and Multi-Output Labels. **CHI 2022** (1/25 best papers)]



Shelf Enable and disable different dimensions of the data. The order of dimension defines the nesting level.

Where Condition the confusion matrix on the value of a given label.

(A) **Normalize** confusions and display performance **metrics**

(B) Interactively traverse and compare **hierarchical labels**

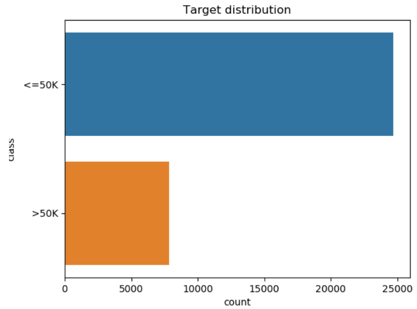
(C) Transform and visualize high-dimensional **multi-output labels**

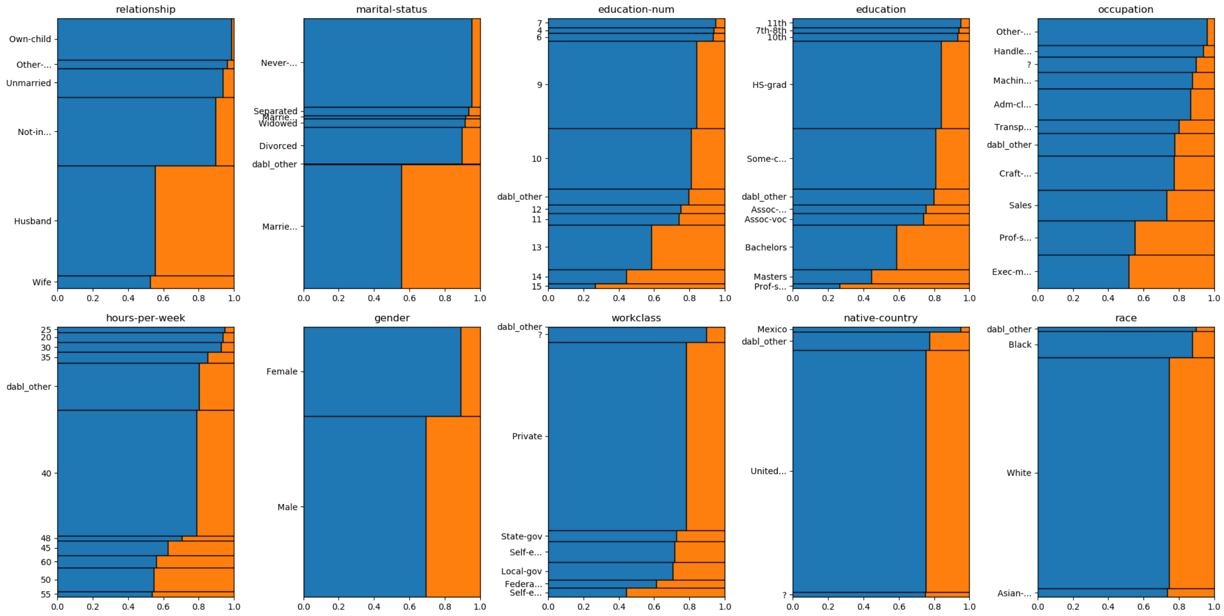
- Transform multi-label data:
  conditioning, marginalization (aggregation), and nesting

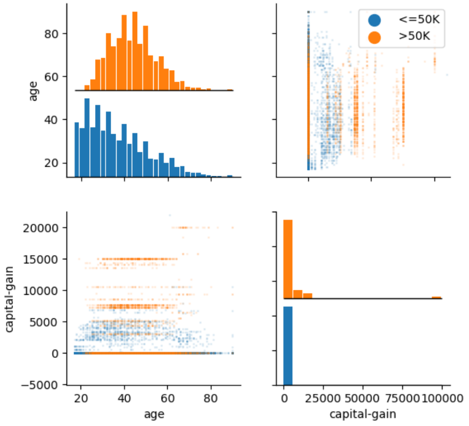
Excursus: dabl.plot

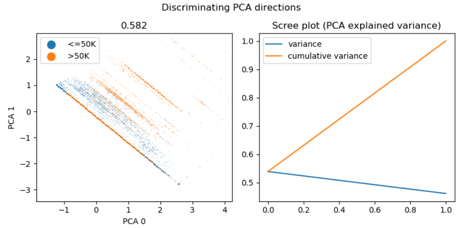
[Andreas Mueller: dabl – Taking the edge off of data science with dabl, **Data Umbrella 2022**, https://www.youtube.com/watch?v=h92RMJi4mRM]

```
# adult dataset (>50K vs <=50K income)
data = pd.read_csv("adult.csv")
plot(data, "income")
```









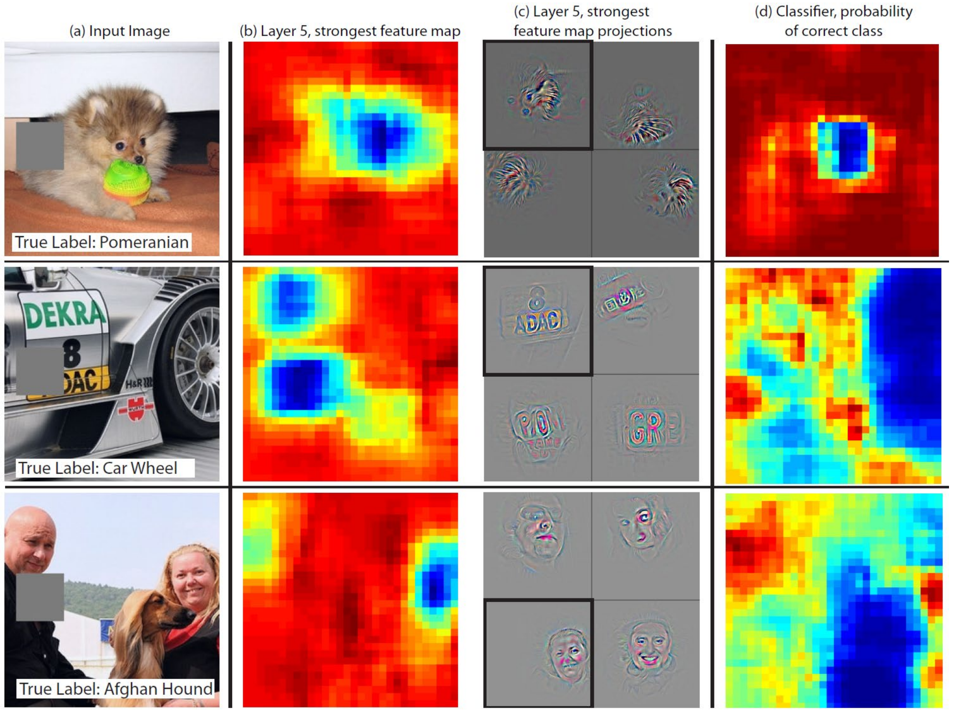
[https://amueller.github.io/dabl/dev/auto_examples/plot/plot_adult.html]

(mosaic plots)

Occlusion-Based Explanations

- **Occlusion Explanations**
  - Slide gray square over inputs
  - Measure how feature maps and classifier output changes

[Matthew D. Zeiler, Rob Fergus: Visualizing and Understanding Convolutional Networks. **ECCV 2014**]


(a) Input Image
(b) Layer 5, strongest feature map
(c) Layer 5, strongest feature map projections
(d) Classifier, probability of correct class
True Label: Pomeranian
True Label: Car Wheel
True Label: Afghan Hound

- **Incremental Computation of Occlusion Explanations**
  - View CNN as white-box operator graph and operators as views
  - Materialize intermediate tensors and apply incremental view maintenance

[Supun Nakandala, Arun Kumar, and Yannis Papakonstantinou: Incremental and Approximate Inference forFaster Occlusion-based Deep CNN Explanations, **SIGMOD 2019**]

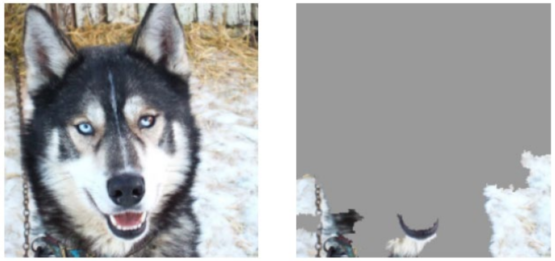
SIGMOD 2020 Research Highlight

Example Model Anomalies

“silent but severe problems”

- **#1 Wolf Detection based on snow cover**

[Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin: Why Should I Trust You?: Explaining the Predictions of Any Classifier, **KDD 2016**]

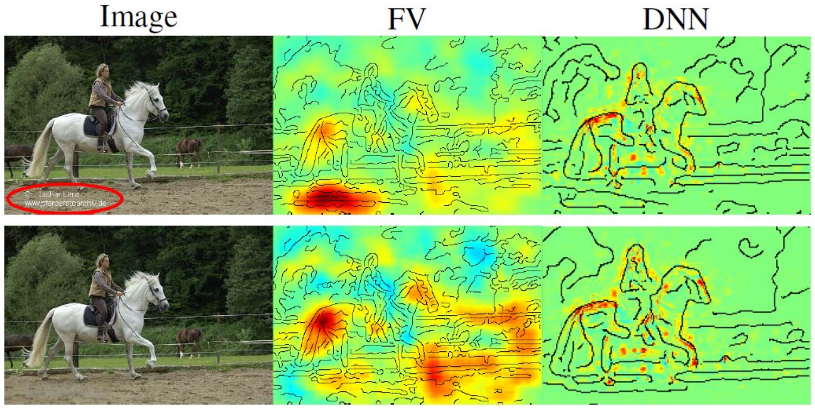


(a) Husky classified as wolf (b) Explanation

12/27 → 25/27

- **#2 Horse Detection based on image watermarks**
  - Layer-wise relevance propagation

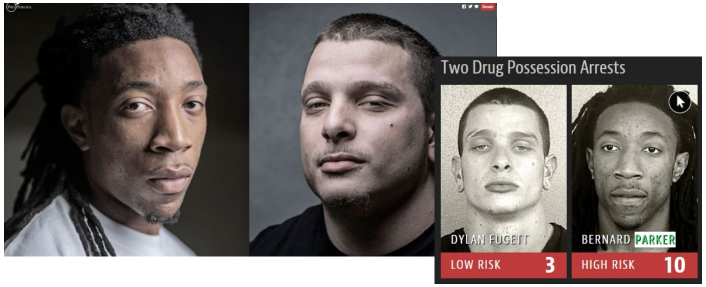
[Sebastian Lapuschkin et al.: Analyzing Classifiers: Fisher Vectors and Deep Neural Networks, **CVPR 2016**]



- **#3 Race-biased Jail Risk Assessment**

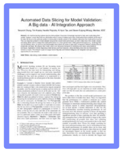
#BlackLivesMatter

[Julia Angwin et al: Machine Bias – There’s software used across the country to predict future criminals. And it’s biased against blacks, **2016**, https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing]



SliceFinder

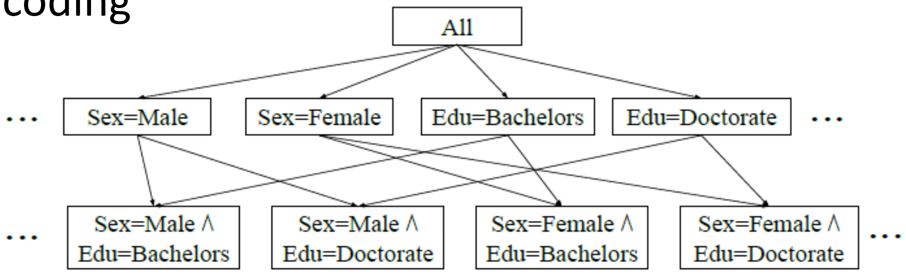
[Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, Steven Euijong Whang: Automated Data Slicing for Model Validation: A Big Data - AI Integration Approach. **ICDE2019/TKDE2020**]



- **Problem Formulation**
  - Data slice: S^{DG} := D=PhD AND G=female (subsets of features)
  - **Find top-k data slices where model performs worse** than average
  - **Ordering by**
    - Increasing number of literals,
    - Decreasing slice size, and decreasing effect size (difference S vs $\neg S$)
  - **Subject to:** minimum effect size threshold T, statistical significance (Welch's t-test), a dominance constraint (no coarser slice satisfies 1 and 2) via

"find largest error vs find large slices"

- **Existing Algorithms**
  - Preparation: Binning + One-Hot Encoding
  - **#1** Clustering → slices
  - **#2** Decision tree training
  - **#3** **Lattice search** with heuristic, level-wise termination



All

… Sex=Male | Sex=Female | Edu=Bachelors | Edu=Doctorate …

… Sex=Male ∧ Edu=Bachelors | Sex=Male ∧ Edu=Doctorate | Sex=Female ∧ Edu=Bachelors | Sex=Female ∧ Edu=Doctorate …

SliceLine for Model Debugging

[**Credit:** sliceline, Silicon Valley, HBO]

- **Problem Formulation**
  - Intuitive slice scoring function
  - Exact top-k slice finding
  - $|S| \geq \sigma \land sc(S) > 0$
  - $\alpha \in (0,1]$

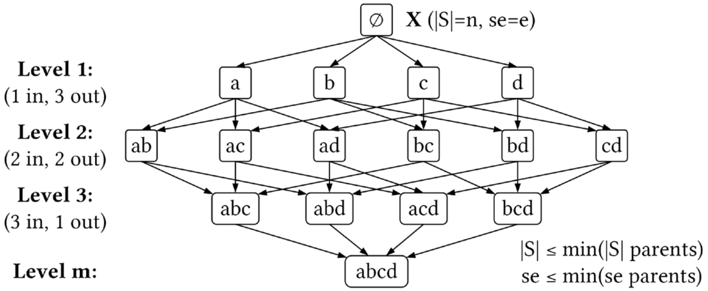
$$sc = \alpha\left(\frac{\bar{e}(S)}{\bar{e}(X)} - 1\right) - (1-\alpha)\left(\frac{|X|}{|S|} - 1\right)$$

$$= \alpha\left(\frac{|X|}{|S|} \cdot \frac{\sum_{i=1}^{|S|} es_i}{\sum_{i=1}^{|X|} e_i} - 1\right) - (1-\alpha)\left(\frac{|X|}{|S|} - 1\right)$$

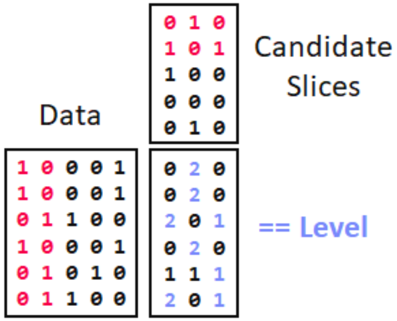
slice error

slice size

- **Properties & Pruning**
  - Monotonicity of slice sizes, errors
  - Upper bound sizes/errors/scores → pruning & termination



- **Linear-Algebra-based Slice Finding**
  - Recoded matrix **X**, error vector e
  - Vectorized implementation in linear algebra (join & eval via sparse-sparse matrix multiply)
  - Local and distributed task/data-parallel execution



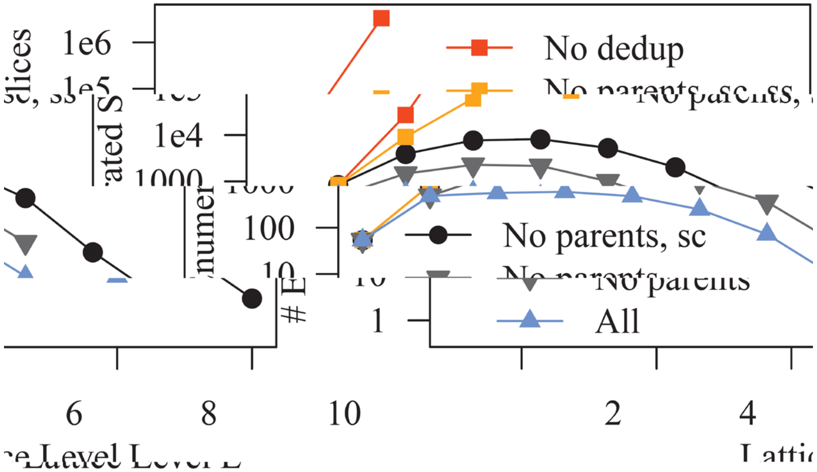
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

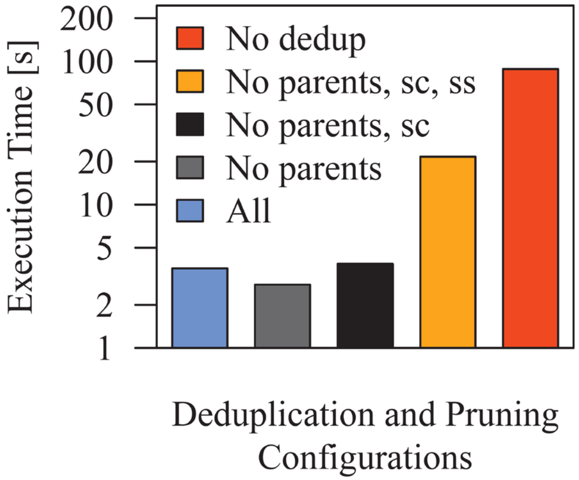
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 2 & 0 \\ 0 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 1 \\ 2 & 0 & 1 \end{bmatrix}$$

SliceLine – Experiments

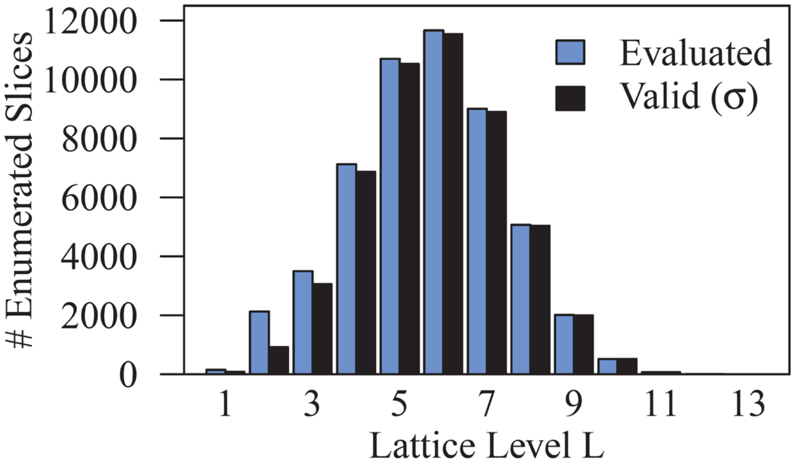
[Svetlana Sagadeeva, Matthias Boehm: SliceLine: Fast, Linear-Algebra-based Slice Finding for ML Model Debugging, **SIGMOD 2021**]

- **Salaries 2x2**





- **Adult**



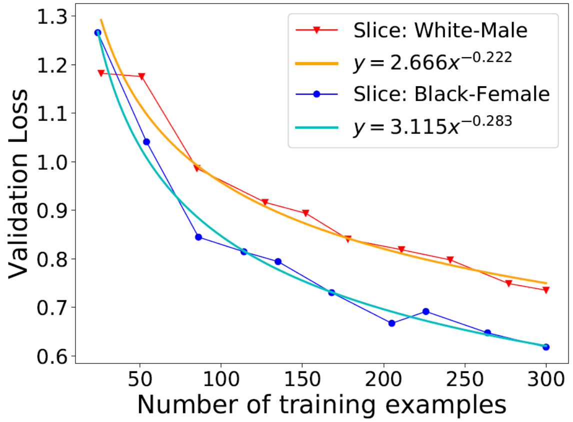
Effective Pruning (#evaluated close to #valid)

Practical Performance (39s until termination at level 12)

Slice Tuner

[Ki Hyun Tae, Steven Euijong Whang: Slice Tuner: A Selective Data Acquisition Framework for Accurate and Fair Machine Learning Models, **SIGMOD 2021**]

- **Motivation**
  - Root cause of unfairness: **bias in training data**
  - **Selective Data Acquisition** for model accuracy and fairness
  - Different slices w/ different learning curves → **Learning curve fitting**



- **Problem Formulation**

Convex optimization problem

$$\overbrace{\min \sum_{i=1}^{n} b_i(|s_i| + d_i)^{-a_i}}^{\text{Minimize total loss of slices}} + \lambda \overbrace{\sum_{i=1}^{n} \max\left\{0, \frac{b_i(|s_i| + d_i)^{-a_i}}{A} - 1\right\}}^{\text{Penalize underperforming slices}}$$

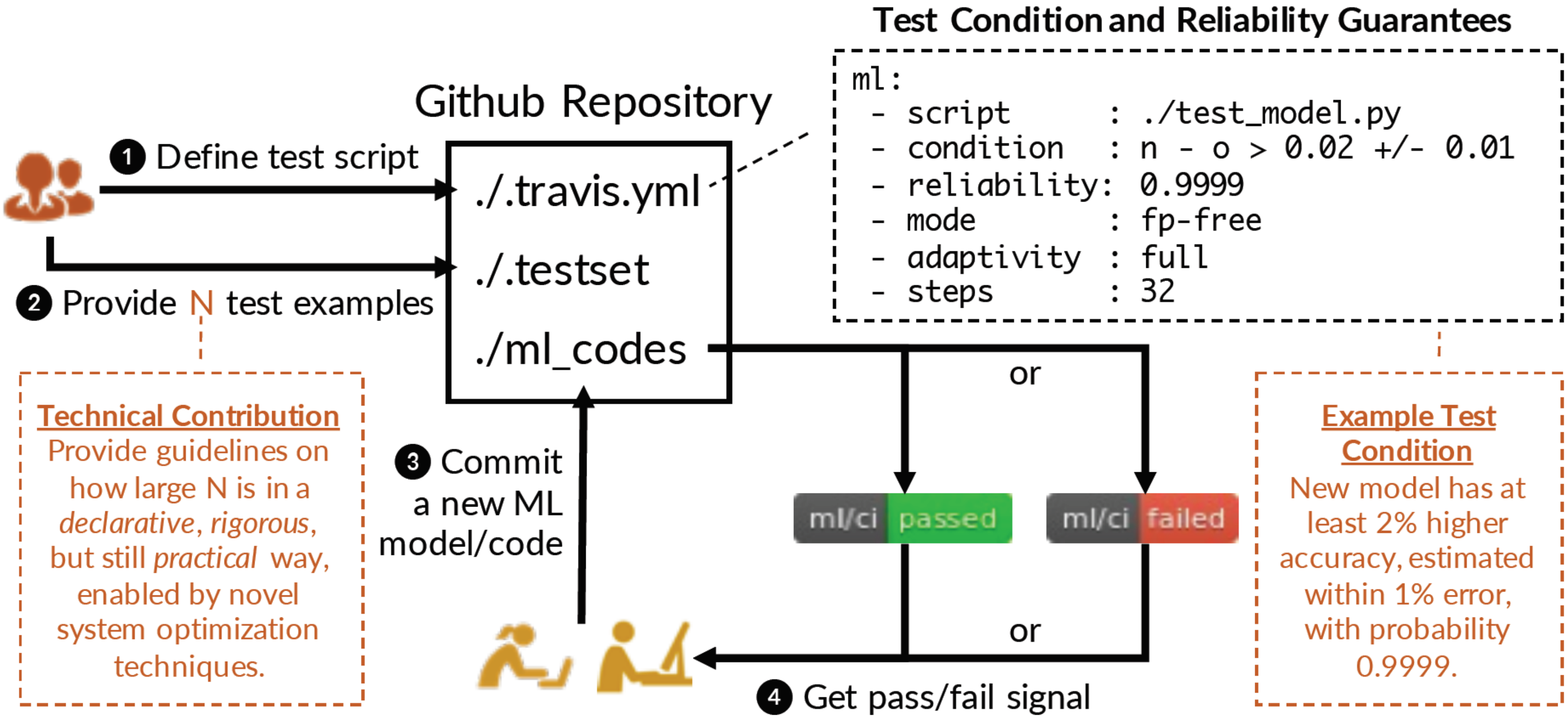
$$\text{subject to} \sum_{i=1}^{n} C(s_i) \times d_i = B$$

Budget of acquisition costs

Continuous Integration

- **System Architecture**
  ease.ml/ci

[Cedric Renggli, Bojan Karlaš, Bolin Ding, Feng Liu, Kevin Schawinski, Wentao Wu, Ce Zhang: Continuous Integration of Machine Learning Models with ease.ml/ci: Towards a Rigorous Yet Practical Treatment, **SysML 2019**]



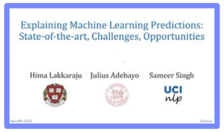
```
ml:
 - script      : ./test_model.py
 - condition   : n - o > 0.02 +/- 0.01
 - reliability: 0.9999
 - mode        : fp-free
 - adaptivity  : full
 - steps       : 32
```

Technical Contribution
Provide guidelines on how large N is in a *declarative*, *rigorous*, but still *practical* way, enabled by novel system optimization techniques.

Example Test Condition
New model has at least 2% higher accuracy, estimated within 1% error, with probability 0.9999.

Explainability

[Hima Lakkaraju, Julius Adebayo, Sameer Singh: Explaining Machine Learning Predictions: State-of-the-art, Challenges, and Opportunities, **NeurIPS 2020** Tutorial, https://explainml-tutorial.github.io/neurips20]



Explaining Machine Learning Predictions: State-of-the-art, Challenges, Opportunities

Hima Lakkaraju Julius Adebayo Sameer Singh

NeurIPS 2020 Tutorial

- **Motivation**
  - Explain predictions via inputs for **model understanding** & **transparency**
  - Utilize model debugging and other tools
- **#1 Interpretable Models**
  - Linear models, tree-based models, rule-based models
  - Weights and decision rules

Interpretability ← → Accuracy

Prefer simpler models if accuracy sufficient!

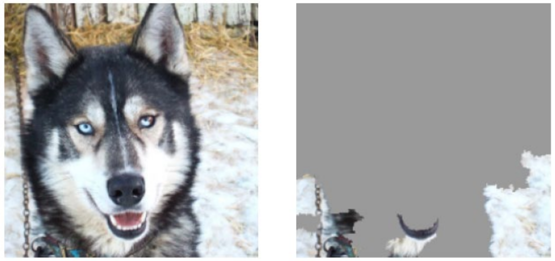
- **#2 Post-hoc Explanations**
  - Complex deep neural networks or very large models → black box models
  - Build simple models for explaining **any** complex models
- **Types of Explanations**
  - Model parameters, example predictions, summarization
  - **Most important features**/data, how to flip model predictions

Multi-modal Interpretability:
https://captum.ai/

LIME: Sparse, Linear Explanations

- **LIME Overview**
  - Model agnostic explanations
  - Identify important dimension and present their relative importance
  - **Sample perturbations** of prediction input (e.g., hide parts of image, attribute values)
  - **Locally weighted regression**

[Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin: Why Should I Trust You?: Explaining the Predictions of Any Classifier, **KDD 2016**]



(a) Husky classified as wolf (b) Explanation

- **LIME Objective**
  - Various hyper-parameters
  - Heuristics / HP optimization

$$\xi(x) = \underset{\text{Linear Models}}{\underset{g \in G}{\operatorname{argmin}}} \; \overset{\text{Loss Function}}{\mathcal{L}}(f, g, \underset{\text{Local Kernel}}{\pi_x}) + \overset{\text{Regularizer}}{\Omega}(g)$$

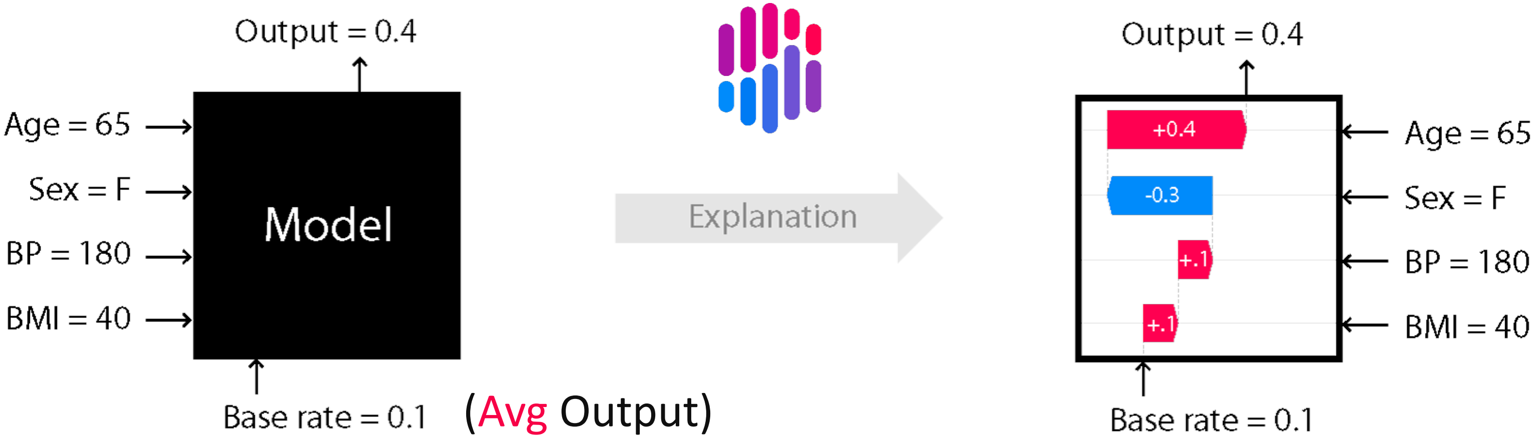
SHAP: Shapley Additive Explanations

[Scott M. Lundberg, Su-In Lee: A Unified Approach to Interpreting Model Predictions. **NIPS 2017**]

- **SHAP Overview**
  - Additive feature importance (local accuracy)
  - Unification of LIME, Shapley sampling/regression values, QII, DeepLIFT, layer-wise relevance propagation, tree interpreter
  - Estimate Shapley values using linear regression

[Scott M. Lundberg: Explainable AI for Science and Medicine, https://www.youtube.com/watch?v=B-c8tIgchu0]

- **SHAP Tooling**



(Avg Output)

[https://shap.readthedocs.io/en/latest/index.html]

Marginal contributions

Model Bias & Fairness

Focus on Applications, Fairness, Ethics, Responsibility
Fairness Metrics and Constraints

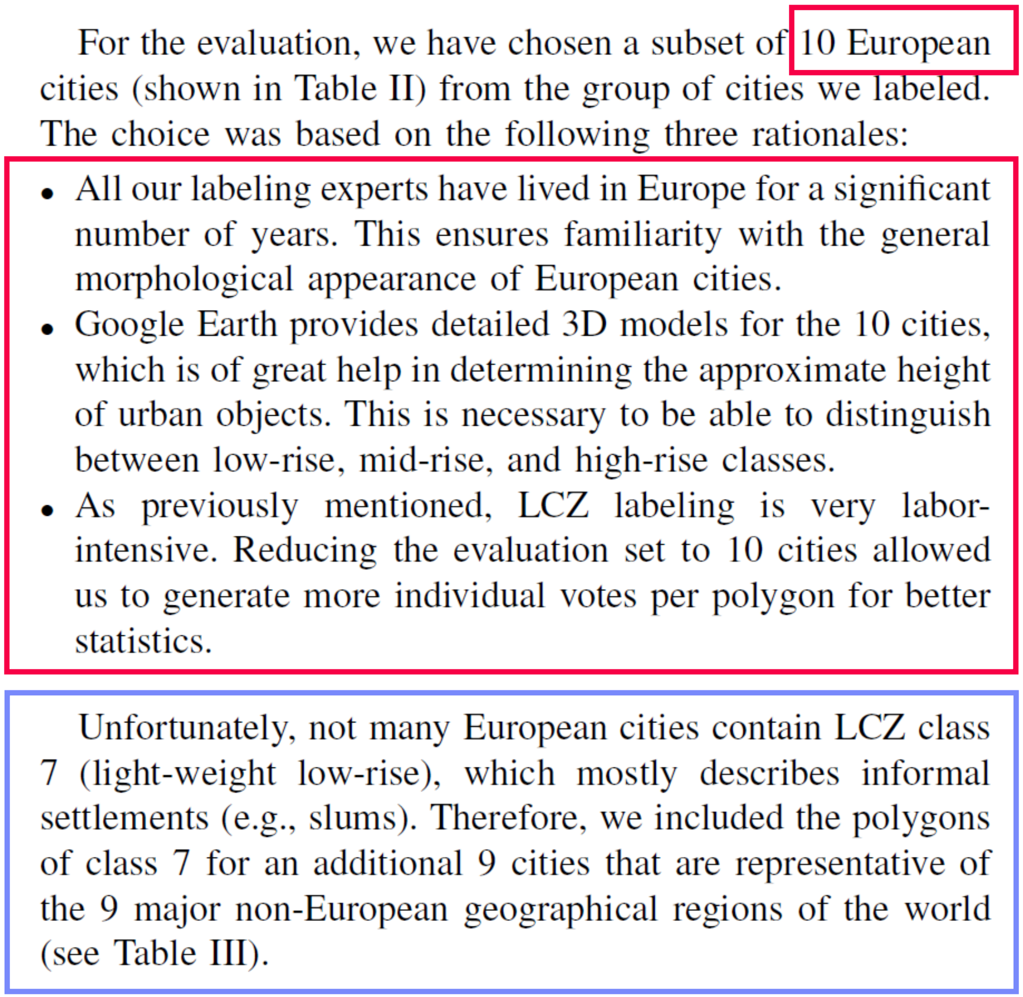
Employs Model Debugging & Explainability Techniques

Sources of Bias

- **Environment**
  - **Selection Bias:** Differences in study participation, data availability, and measurement effort
  - Test environment, project team, cultural context → **different context**
- **Data Collection**
  - **Sample Bias:** collected data not representative of application
  - **Observer Bias** / **Confirmation Bias:** subjective judgment leaks into measurement and analysis → **transparency and critical feedback**
- **Training Dataset**
  - **Data Bias:** e.g., not missing at random (NMAR) values
  - **Feature Selection Bias:** manual or automatic during data preparation

→ Design ML Systems & applications w/ awareness of potential bias

Excursus: DLR Earth Observation Use Case, cont.



For the evaluation, we have chosen a subset of 10 European cities (shown in Table II) from the group of cities we labeled. The choice was based on the following three rationales:

- All our labeling experts have lived in Europe for a significant number of years. This ensures familiarity with the general morphological appearance of European cities.
- Google Earth provides detailed 3D models for the 10 cities, which is of great help in determining the approximate height of urban objects. This is necessary to be able to distinguish between low-rise, mid-rise, and high-rise classes.
- As previously mentioned, LCZ labeling is very labor-intensive. Reducing the evaluation set to 10 cities allowed us to generate more individual votes per polygon for better statistics.

Unfortunately, not many European cities contain LCZ class 7 (light-weight low-rise), which mostly describes informal settlements (e.g., slums). Therefore, we included the polygons of class 7 for an additional 9 cities that are representative of the 9 major non-European geographical regions of the world (see Table III).

[Xiao Xiang Zhu et al: So2Sat LCZ42: A Benchmark Dataset for the Classification of Global Local Climate Zones. **GRSM 2020**]

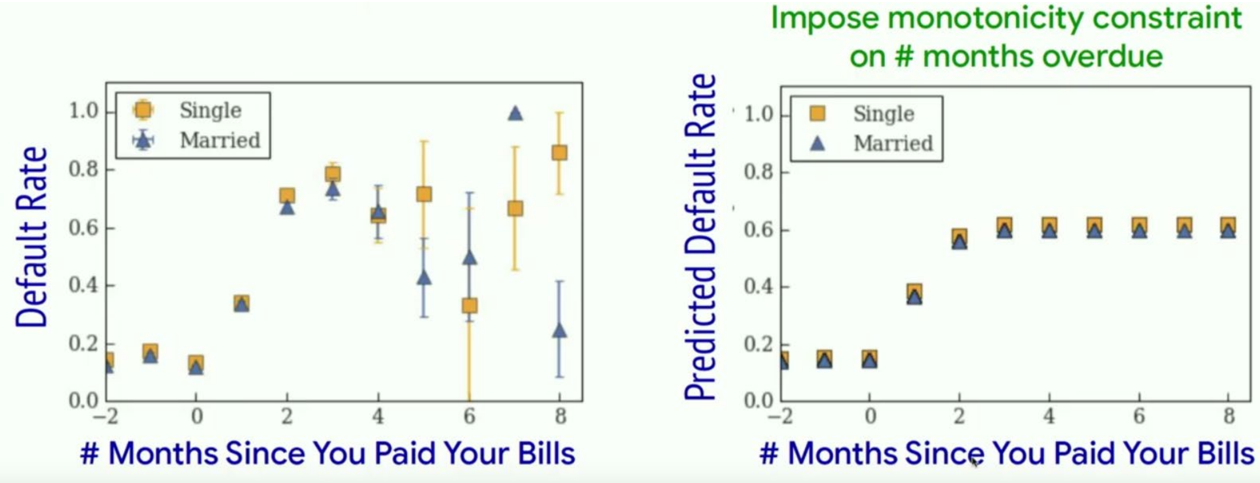
Environment / **Context**
→ **Biased Data Collection**

→ **Awareness and Conscious Bias Mitigation**
→ **Remaining Bias?**

Debugging Bias and Fairness

- **Fairness**
  - Validate and ensure fairness with regard to sensitive features (unbiased)
  - Use occlusion and saliency maps to characterize and compare groups

- **Enforcing Fairness**
  - Use constraints to enforce certain properties (e.g., monotonicity, smoothness)
  - Example: late payment → credit score



[Maya Gupta: How Do We Make AI Fair? **SysML 2019**]

Group Fairness Constraints

[H. Zhang et al: OmniFair: A Declarative System for Model-Agnostic Group Fairness in Machine Learning, **SIGMOD 2021**]

- **#1 Statistical Parity**
  - Independence of model from groups
  - **Equal probability outcome** across groups

$$\forall g_i, g_j \in G: \\ P(h=1|g_i) \approx P(h=1|g_j)$$

- **#2 False Positive Rate Parity**
  - Independence of model from groups
  - **Conditioned on true label y=0**

$$\forall g_i, g_j \in G: \\ P(h=1|g_i, y=0) \\ \approx P(h=1|g_j, y=0)$$

#2+#3 Equalized Odds

- **#3 False Negative Rate Parity**
  - Independence of model from groups
  - **Conditioned on true label y=1**

$$\forall g_i, g_j \in G: \\ P(h=0|g_i, y=1) \\ \approx P(h=0|g_j, y=1)$$

- **#4 False Omission Rate Parity**
  - Independence of true labels from groups
  - **Conditioned on negative prediction h=0**

$$\forall g_i, g_j \in G: \\ P(y=1|g_i, h=0) \\ \approx P(y=1|g_j, h=0)$$

Ensuring Fairness

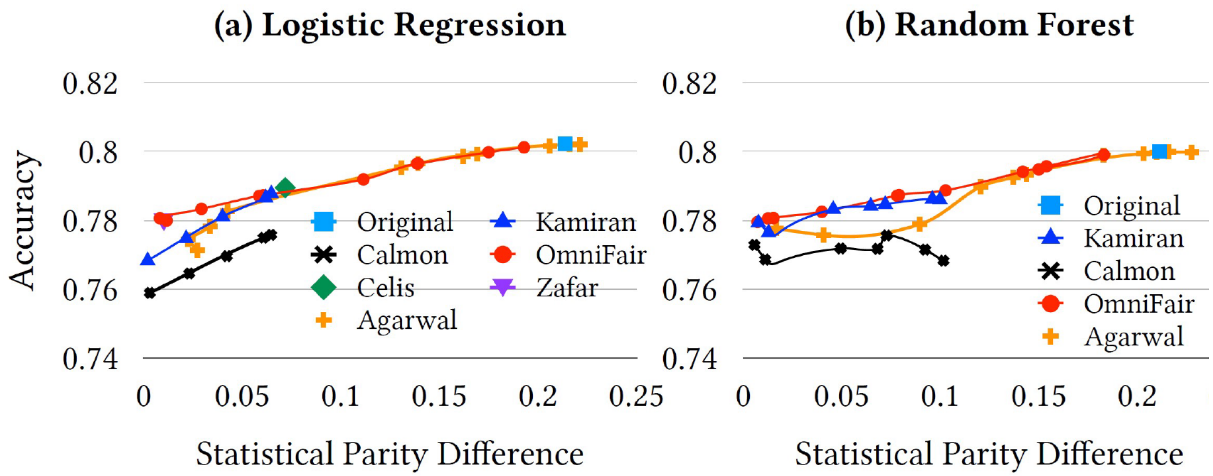
[H. Zhang et al: OmniFair: A Declarative System for Model-Agnostic Group Fairness in Machine Learning, **SIGMOD 2021**]

- **Problem Formulation**
  - A **fairness specification** is given by a triplet (g, f, ε) and induces (|g(D)|choose 2) **fairness constraints** on pairs of groups
  - A fairness specification is satisfied by a classifier h on D iff all induced fairness constraints are satisfied, i.e., ∀gi,gj ∈ g(D), |f(h,gi)−f(h,gj)| ≤ ε
  - **Unconstrained optimization problem**

max accuracy
s.t. fairness

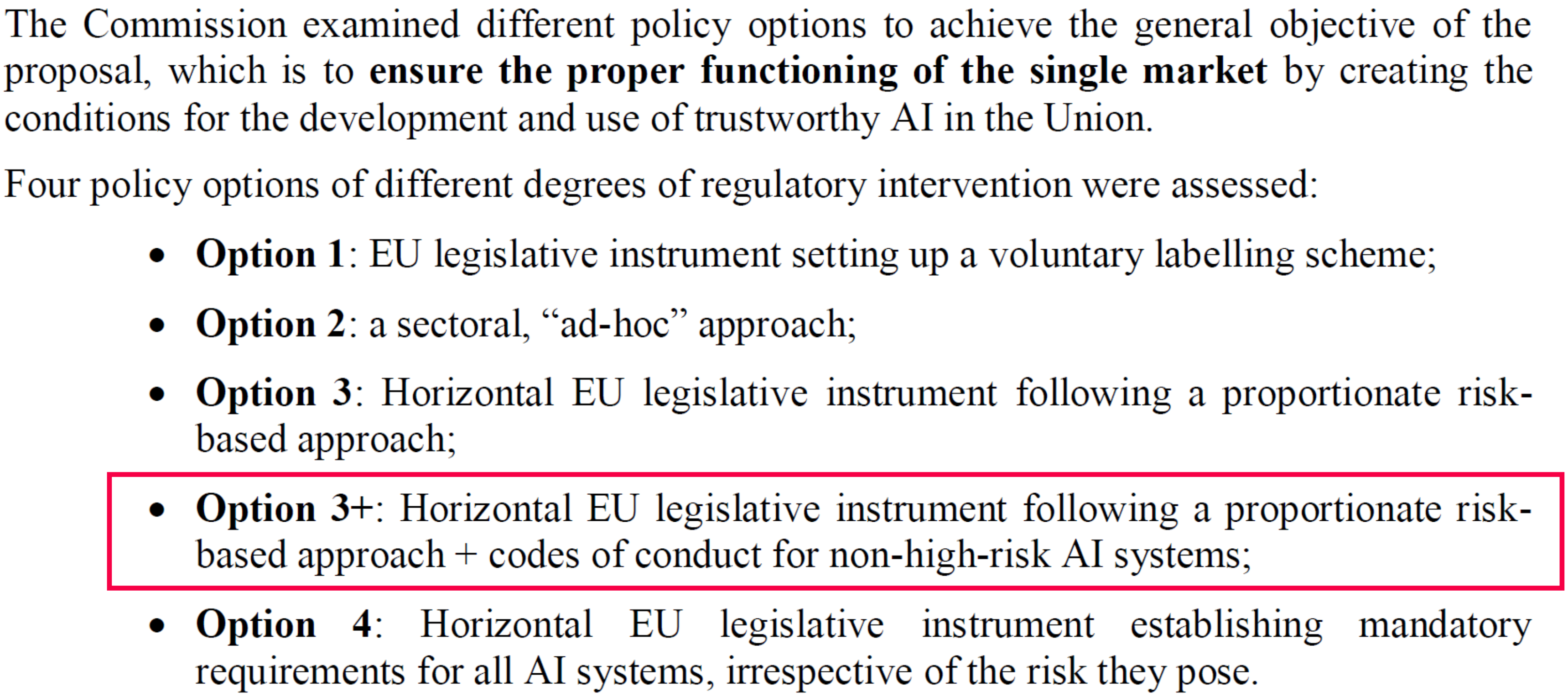
max accuracy
+ fairness

- **Results**
  - Adult dataset
  - Model-agnostic
  - Similar Accuracy



Excursus: EU Policy

[European Commission: LAYING DOWN HARMONISED RULES ON ARTIFICIAL INTELLIGENCE (ARTIFICIAL INTELLIGENCE ACT) AND AMENDING CERTAIN UNION LEGISLATIVE ACTS, 04/2021]



The Commission examined different policy options to achieve the general objective of the proposal, which is to **ensure the proper functioning of the single market** by creating the conditions for the development and use of trustworthy AI in the Union.

Four policy options of different degrees of regulatory intervention were assessed:

- **Option 1**: EU legislative instrument setting up a voluntary labelling scheme;
- **Option 2**: a sectoral, “ad-hoc” approach;
- **Option 3**: Horizontal EU legislative instrument following a proportionate risk-based approach;
- **Option 3+**: Horizontal EU legislative instrument following a proportionate risk-based approach + codes of conduct for non-high-risk AI systems;
- **Option 4**: Horizontal EU legislative instrument establishing mandatory requirements for all AI systems, irrespective of the risk they pose.

➔ “The **preferred option is option 3+**, a regulatory framework for high-risk AI systems only, with the possibility for […] non-high-risk AI systems to follow a code of conduct.”

Summary and Q&A

- **Model Selection Techniques**
- **Model Management & Provenance**
- **Model Debugging and Explainability**
- **Model Bias & Fairness Constraints**