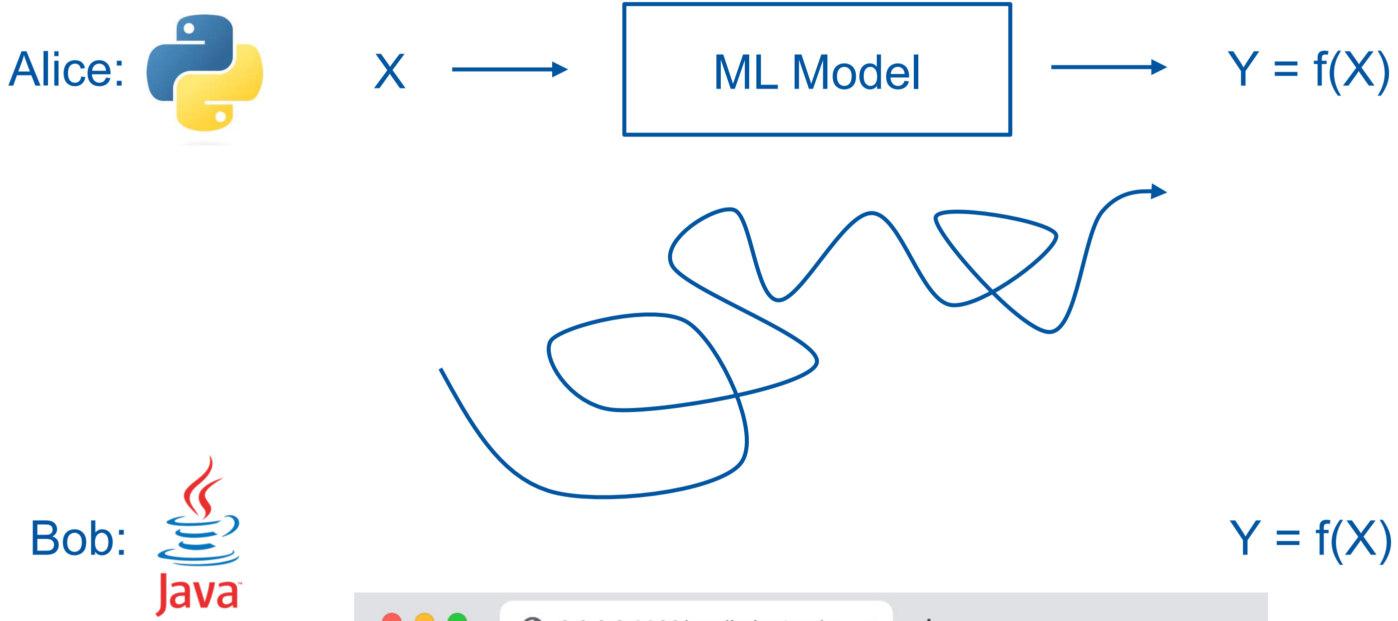
Machine Learning Operations - MLOps

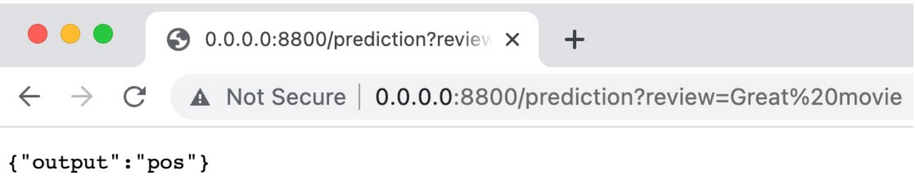
Getting from Good to Great

Slide credit: Michal Maciejewski, PhD

14th Inverted CERN School of Computing

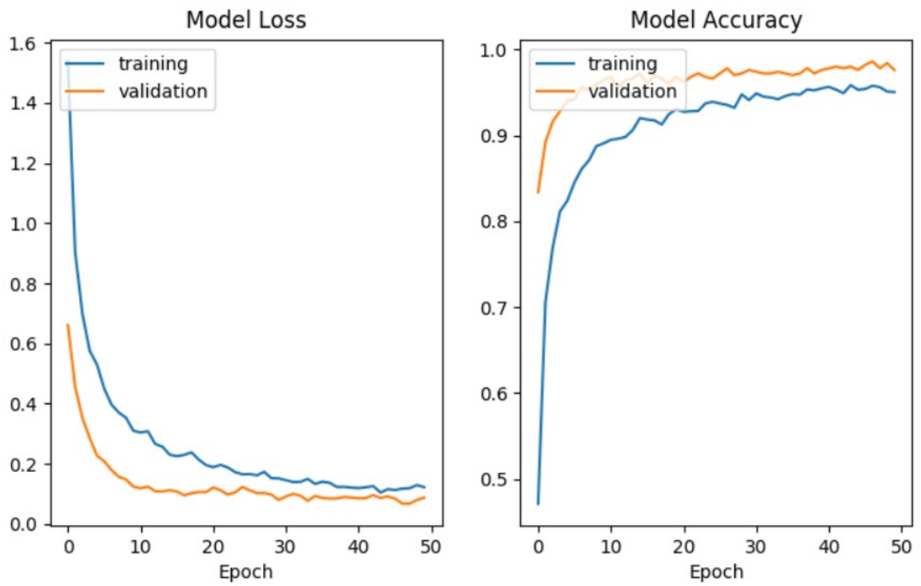
Acknowledgements: Dejan Golubovic, Ricardo Rocha, Christoph Obermair, Marek Grzenkowicz





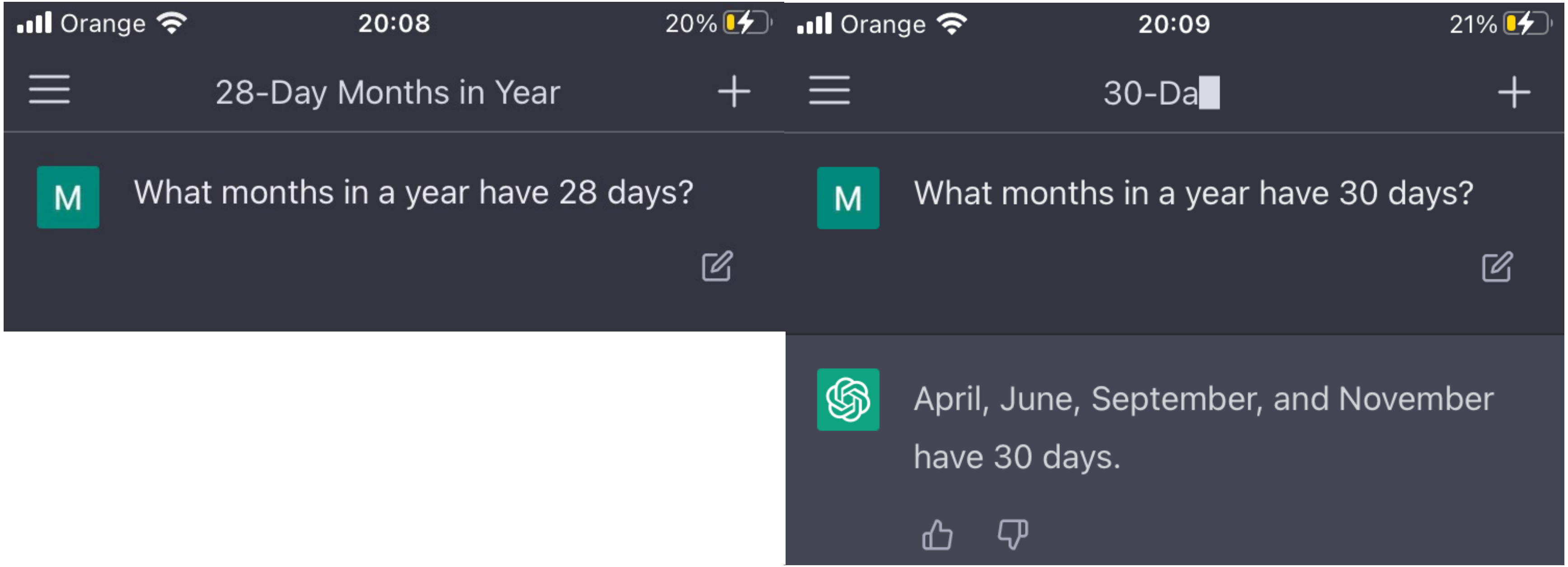
Let’s share our model with users *aka* let’s put it into production!

What Has to Go Right?



What is needed for an ML model to perform well in production?

What Can Go Wrong?



Concept and data drifts are one of the main challenges of production ML systems!

MLOps is about maintaining the trained model performance in production.*
The performance may degrade due to factors outside of our control
so we ought to monitor the performance and if needed, roll out a new model to users.

**model performance = accuracy, latency, jitter, etc.*

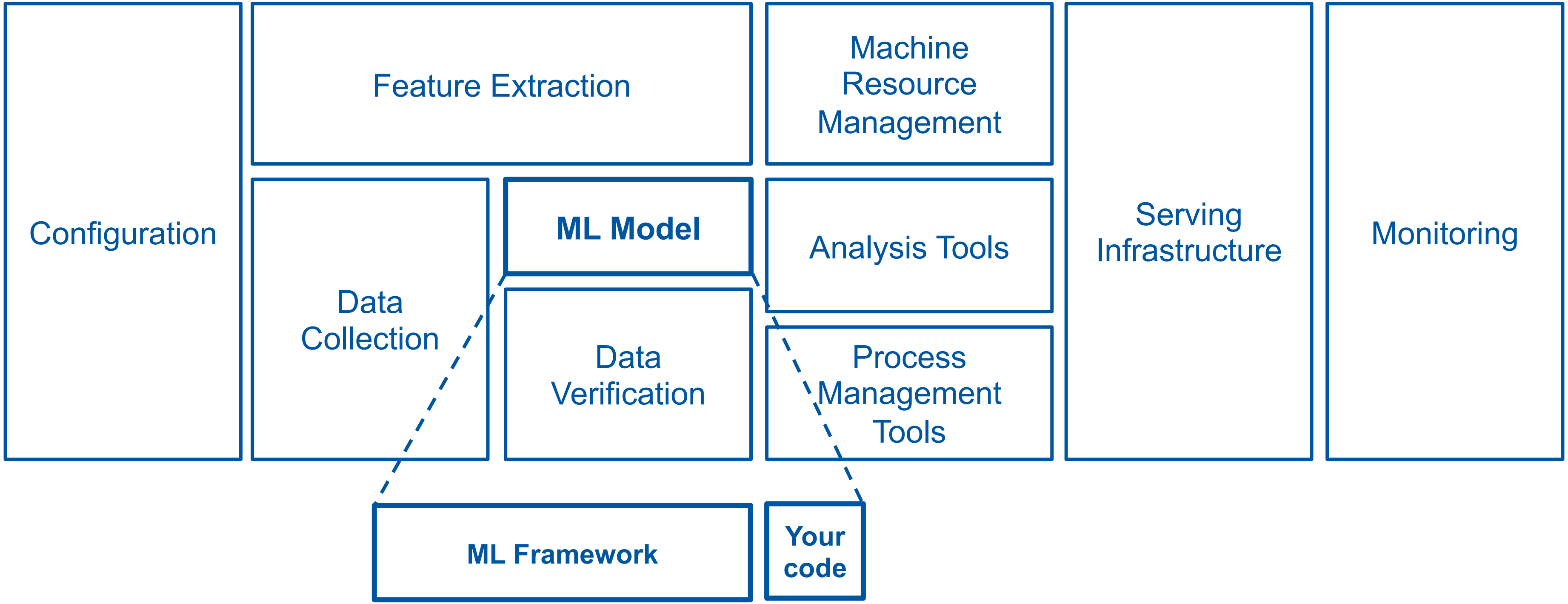
ML Model = Data + Code

MLOps = ML Model + Software

+ Algorithm
+ Weights
+ Hyperparameters

+ Scripts
+ Libraries
+ Infrastructure
+ DevOps

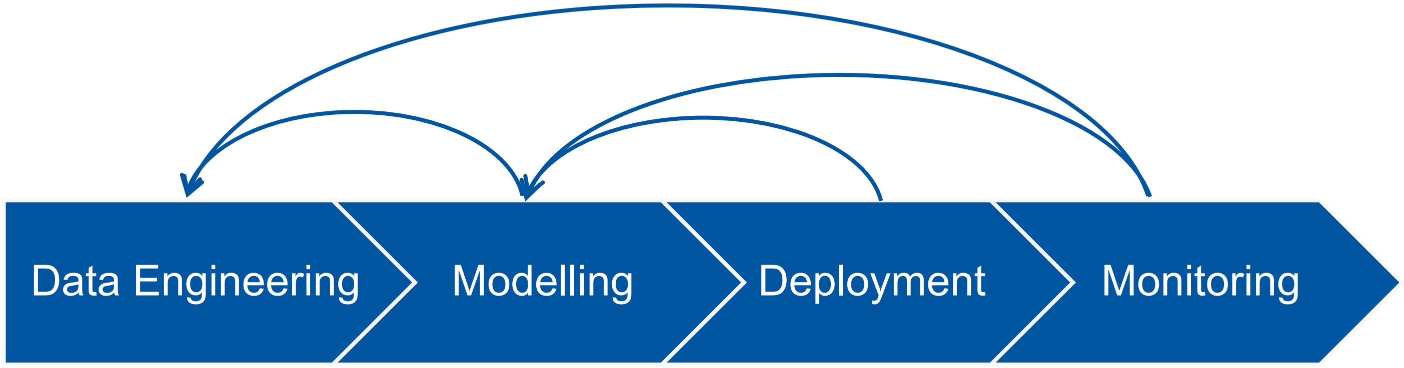
MLOps = ML Model + Software



Good news: most of these components come as ready-to-use frameworks

D. Sculley et. al. *Hidden Technical Debt in Machine Learning Systems, NIPS 2015*

MLOps Pipeline



MLOps is a multi-stage, iterative process.

Data Engineering

Reproducibility

Traceability

Data-driven ML

f() = 

Exploratory Data Analysis

For structured data:

- schema as required tables, columns and datatypes

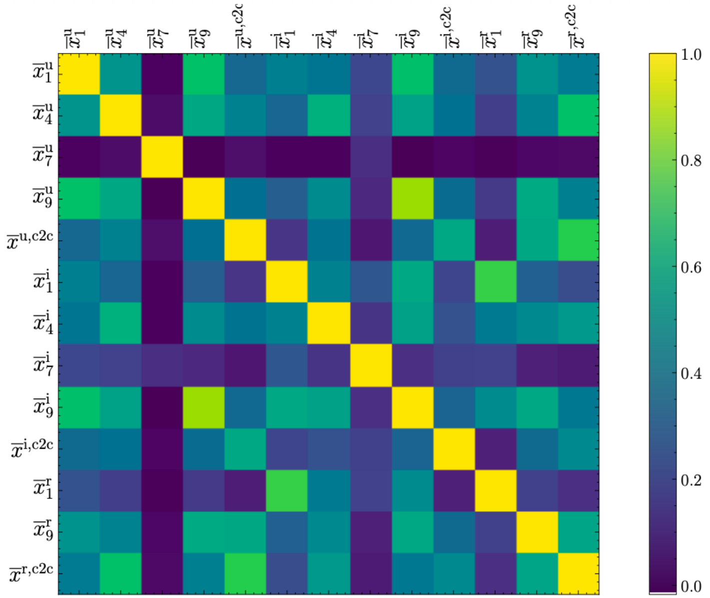
For unstructured data:

- resolution, image extension
- frequency, duration, audio codec

```
DataFrame.corr(method='pearson', min_periods=1, numeric_only=_NoDefault.no_default)
```

Compute pairwise correlation of columns, excluding NA/null values.

[source]



Initial exploration allows indetifying requirements for input data in produciton.

Data Processing Pipeline

Data Ingestion → **Data Validation** → **Data Cleaning** → **Feature Engineering**

Data Ingestion
- Load from file
- Load from db

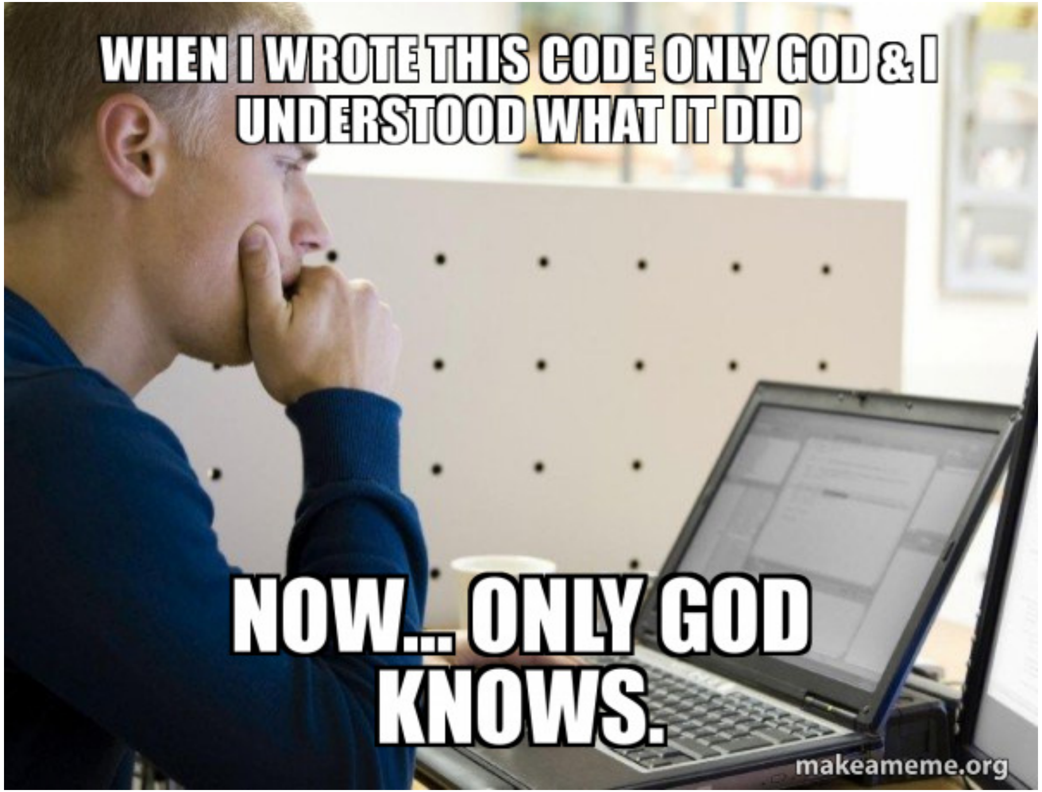
Data Validation
- Schema check
- Audio/video file check

Data Cleaning
- Filling NaNs
- Filtering
- Normalization
- Standardization

Feature Engineering
- Feature selection
- Feature crossover

We need to reproduce some of those steps (e.g. subtracting mean) in production!

Reproducibility



https://sites.google.com/princeton.edu/rep-workshop/

Keeping Track of Data Processing

- Version Input Data – DVC framework
- Version Processing Script - GitLab
- Version Computing Environment - Docker

Data Provenance – where does data come from?
Data Lineage – how data is manipulated?

Notebook Good Practices

Import Libraries

```
In [1]: import plotly.offline as pyo
        # Set notebook mode to work in offline
        pyo.init_notebook_mode()

        import sys
        sys.path.append('..')
        from magnumapi.geometry.CosThetaGeometry import CosThetaGeometry
        from magnumapi.tool_adapters.ansys.AnsysToolAdapter import AnsysToolAdapter
```

```
Analysis executed on 2021-05-26 10:46:25
Loaded MagNum API version 0.0.1
Loaded Tool Adapter version 0.0.1 for ANSYS 2021R1
```

Build Geometry

```
In [ ]: phi_1 = 0.229 * 1.05
```

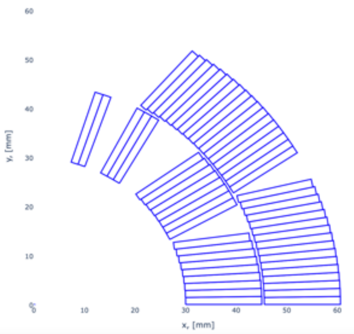
```
In [2]: geometry = CosThetaGeometry.with_roxie_absolute_angle_definition_json('magnet_input_11T.json')
        roxie_df = geometry.to_roxie_df()
        geometry.display_table(roxie_df)
```

Filter... Edit Delete Export to CSV Export to Excel

No	Type	Nco	Radius	Phi	Alpha	Current	Condname	N1	N2	Imag	Turn
1	1	9	30	phi_1	0	11850	FNAL40_NC	2	20	0	0
2	1	8	30	26.5021	28	11850	FNAL40_NC	2	20	0	0
3	1	3	30	55.7611	59	11850	FNAL40_NC	2	20	0	0
4	1	2	30	70.3836	70	11850	FNAL40_NC	2	20	0	0
5	1	16	45.55	0.15	0	11850	FNAL40_NC	2	20	0	0
6	1	18	45.55	30.1226	33	11850	FNAL40_NC	2	20	0	0

Plot Geometry

```
In [3]: geometry.build_blocks()
        geometry.plotly_blocks()
```



- Linear flow of execution
- Little amount of code
- Extract reusable code into a package
- Pre-commit for cleaning notebook before committing to a repository
- Set parameters on top so that notebook can be treated as a function (papermill and scrapbook packages)

It is OK, to do exploratory quick&dirty model development.
Once we start communicating the model outside, we need to clean it!

From Model-driven to Data-driven ML

	Model-driven ML	Data-driven ML
Fixed component	Dataset	Model Architecture
Variable component	Model Architecture	Dataset
Objective	High accuracy	Fairness, low bias
Explainability	Limited	Possible

https://datacentricai.org
https://spectrum.ieee.org/andrew-ng-data-centric-ai

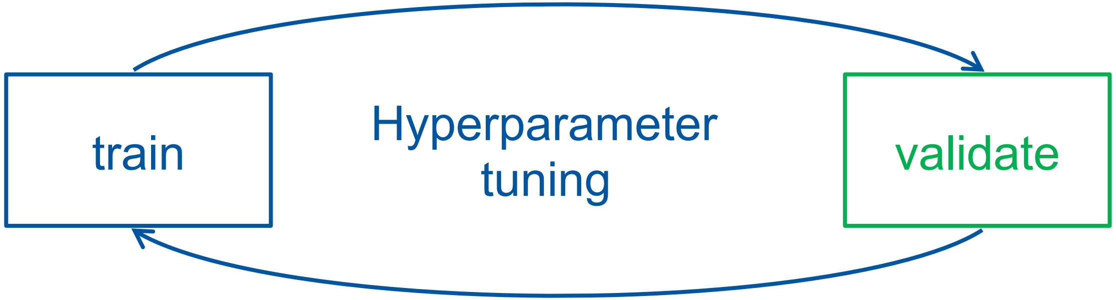
Modelling

Training challenges

Rare events

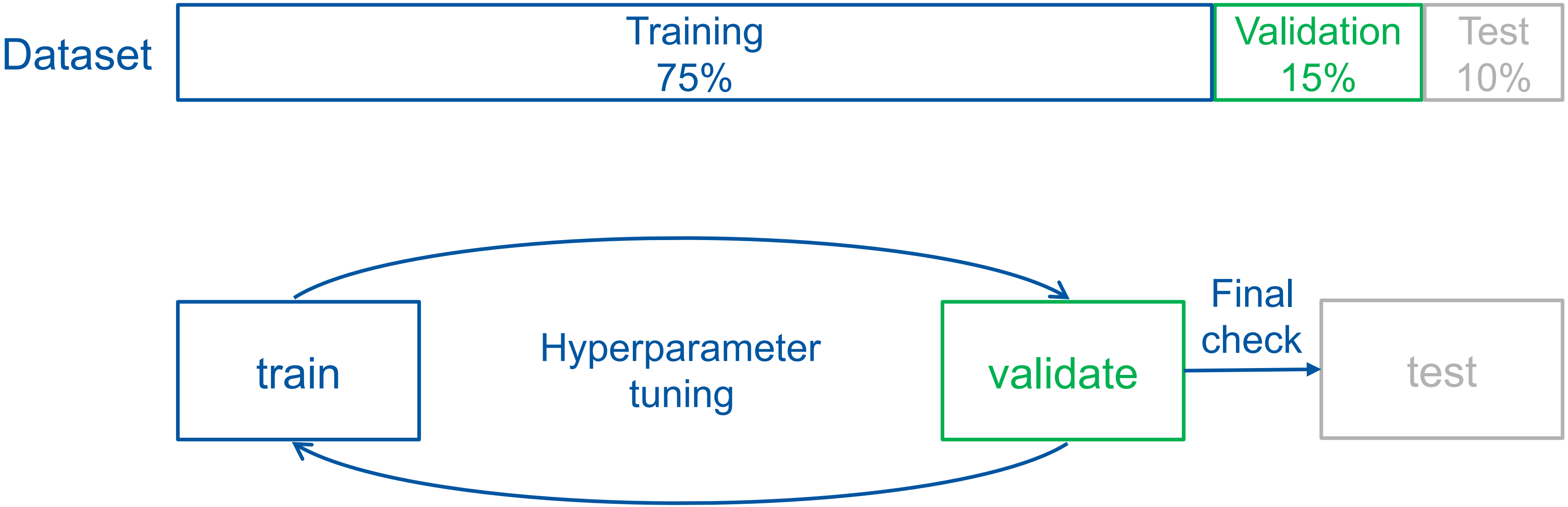
Analyzing results

Selecting Data for Training



With this approach, the model eventually sees the entire dataset.

Selecting Data for Training



Splitting dataset in three allows to perform a final check with unseen data.

Balancing Datasets

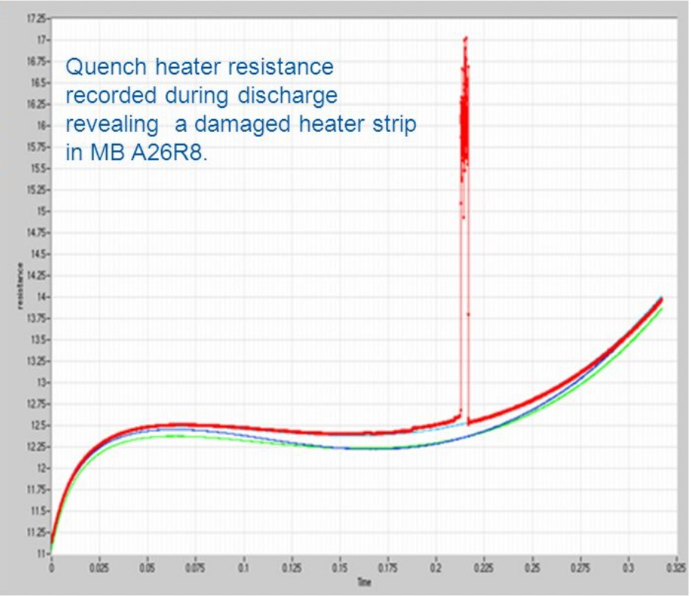
Consider a binary classification problem with a dataset composed of 200 entries.

There are 160 negative examples (no failure) and 40 positive ones (failure).

	Training	Validation	Test
Expected:	75% (120 + 30)	15% (24+6)	10% (16+4)
Random:	75% (131 + 19)	15% (19+11)	10% (10+10)

For continuous values it is important to preserve statistical distribution.
Although for big datasets it is not an issue, it is still a low-hanging-fruit.

Rare Events



There were 3130 healthy signals (Y=False) and 112 faulty ones (Y=True)

C. Obermair, Extension of Signal Monitoring Applications with Machine Learning, Master Thesis, TU Graz
M. Brice, LHC tunnel Pictures during LS2, https://cds.cern.ch/images/CERN-PHOTO-201904-108-15

Rare Events

```python
import pandas as pd


def run_prediction(signal: pd.DataFrame) -> bool:
    return False

```

This *naive* model is guaranteed to achieve 97% average dataset accuracy?!

Rare Events

		Ground truth	
		Y = True	Y = False
Model	Y = True	0 *true positive*	0 *false positive*
	Y = False	112 *false negative*	3130 *true negative*

$$\text{Avg accuracy} = \frac{\text{TN}}{\text{TN} + \text{FN}} = 97\%$$

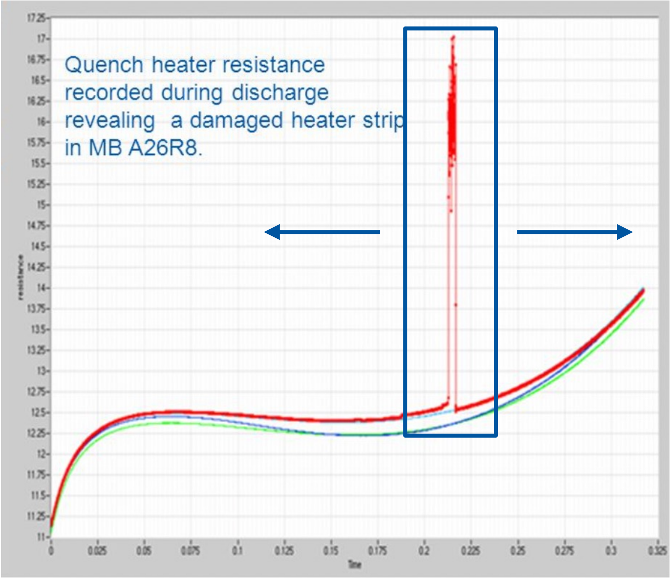
$$\text{Precision} = \frac{TP}{TP + FP} = \frac{0}{0}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{0}{0 + 112} = 0$$

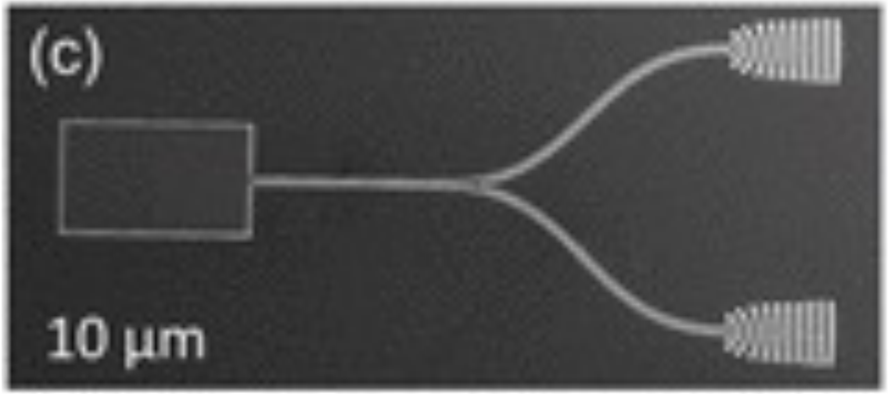
$$\text{F1}_{\text{score}} = \frac{2}{1/\text{Precision} + 1/\text{Recall}}$$

It is a valuable conversation to decide if precision or recall (or both) is more important.

Data Augmentation



New examples obtained by shifting the region left and right



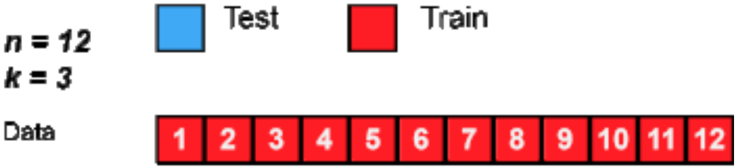
New examples obtained by rotating/shifting/hiding

JH. Kim et al. Hybrid Integration of Solid-State Quantum Emitters on a Silicon Photonic Chip, Nano Letters 2017

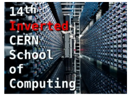
What else can we do?

When one of the values of Y is rare in the population, considerable resources in data collection can be saved by randomly selecting within categories of Y. […]

The strategy is to select on Y by collecting observations (randomly or all those available) for which Y = 1 (the "cases") and a random selection of observations for which Y = 0 (the "controls").



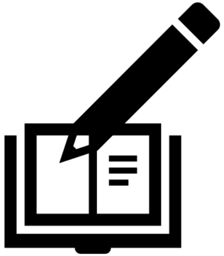
We can also collect more data of particular class (if even possible).

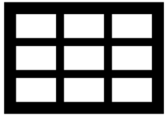


G. King and L. Zeng, "Logistic Regression in Rare Events Data," Political Analysis, p. 28, 2001.
https://en.wikipedia.org/wiki/Cross-validation_(statistics)

Training Tracking

1. Pen & Paper
2. Spreadsheet
3. Dedicated framework
   - Weights and Biases
   - Neptune.ai
   - Tensorflow
   - …





Error Analysis

#	Signal	Noise	Gap in signal	Bias	Wrong sampling
1	Magnet 1	x	x		
2	Magnet 2			x	x
3	Magnet 3	x	x		

Such analysis may reveal issues with labelling or rare classes in data.

For unstructured data, a cockpit could help in analysis.

Useful in monitoring of certain classes of inputs.

Andrej Karpathy
@karpathy

When you sort your dataset descending by loss you are guaranteed to find something unexpected, strange and helpful.

6:23 am · 2 Oct 2020

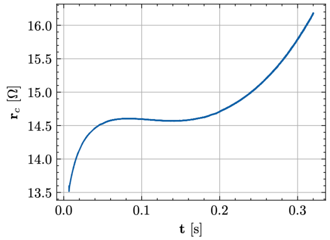
230 Retweets **41** Quote Tweets **2,173** Likes

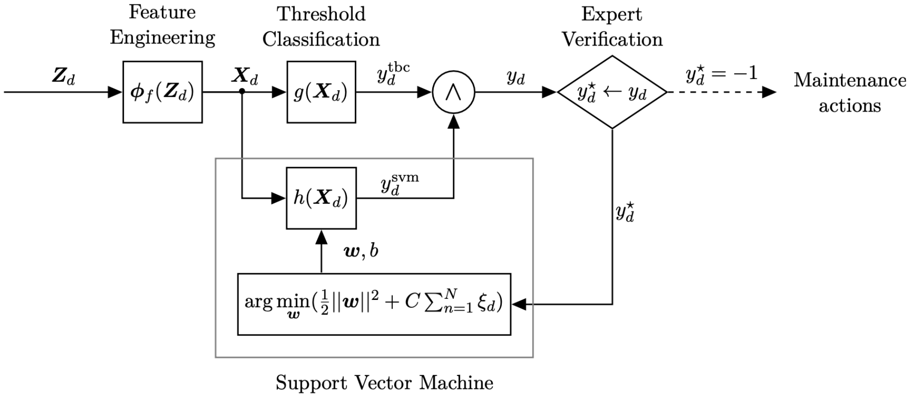
Deployment

Degrees of automation
Modes of deployment
Reproducible environments

Degrees of Automation

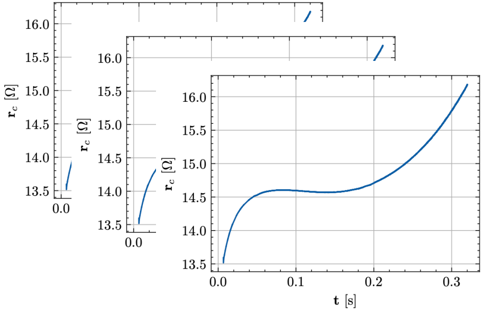
Human inspection → Shadow mode → Human in the loop → Full Automation

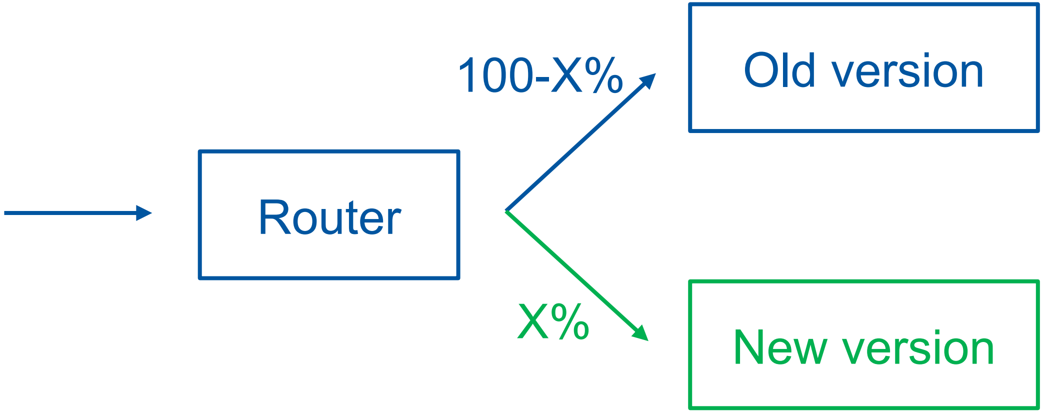




Starting from *Shadow mode* we can collect more training data in production!

Modes of Deployment

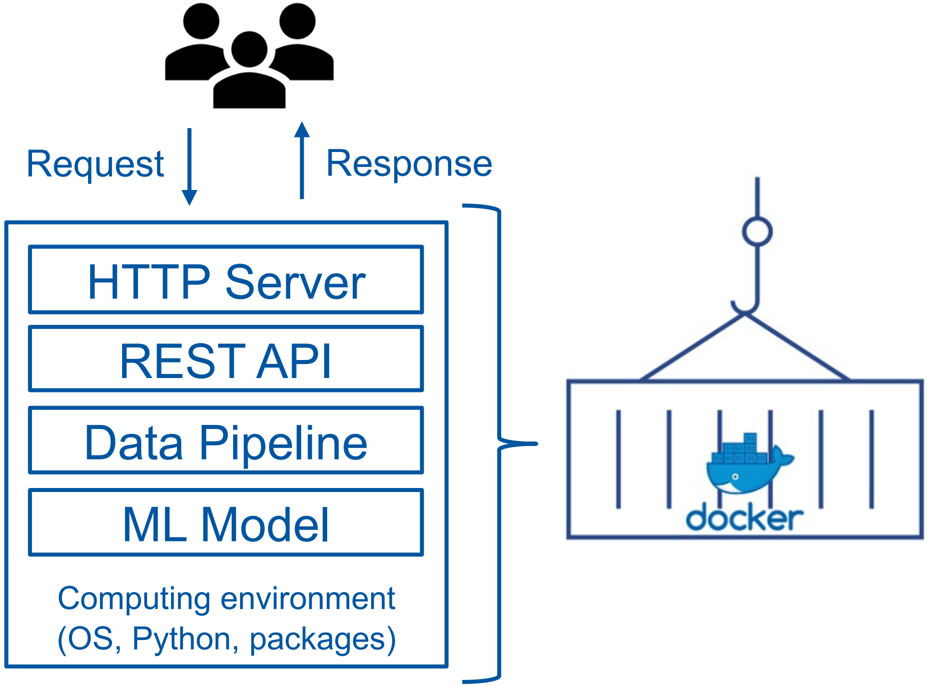




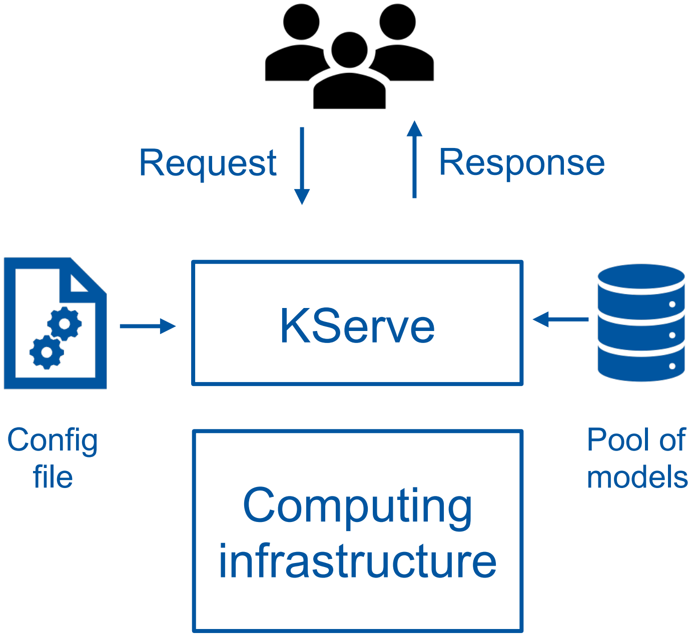
- In **Canary** deployment there is a gradual switch between versions
- In **Blue/green** deployment there is an on/off switch between versions

https://hbr.org/2017/09/the-surprising-power-of-online-experiments
https://en.wikipedia.org/wiki/Blue-winged_parrot

Reproducible Environments



Docker Containers



Serverless compute

We will play with those during the exercise sessions!

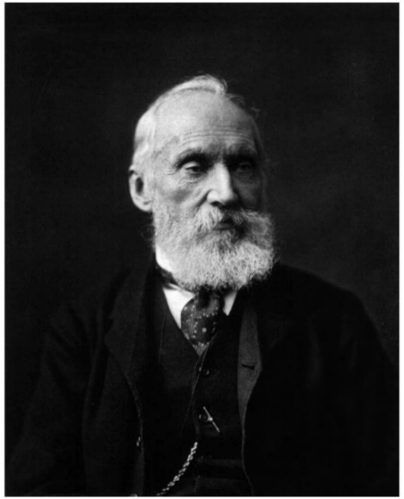
Monitoring

Useful metrics

Relevant frameworks

If you can't measure it,
you can't improve it

William Thomson, Lord Kelvin



14th Inverted CERN School of Computing

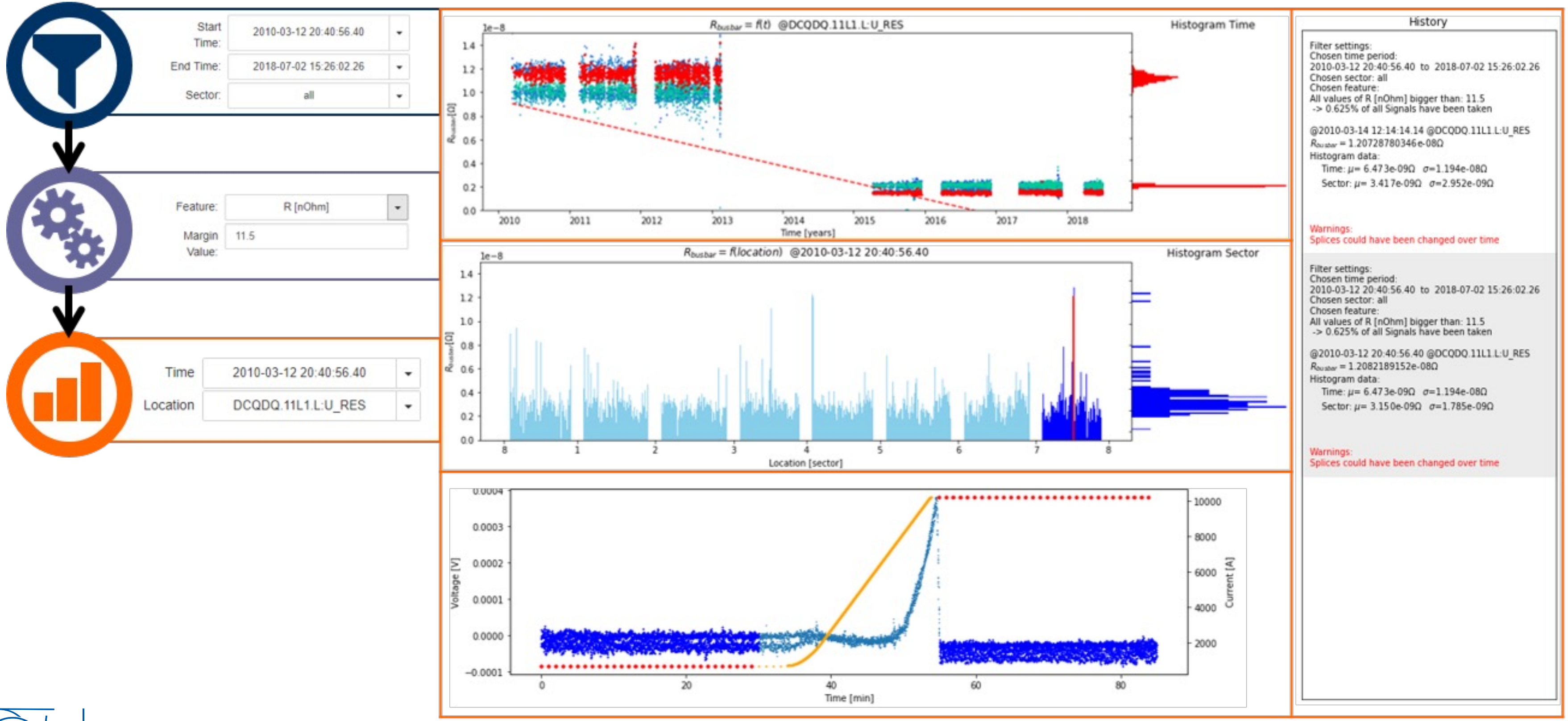
Relevant Metrics

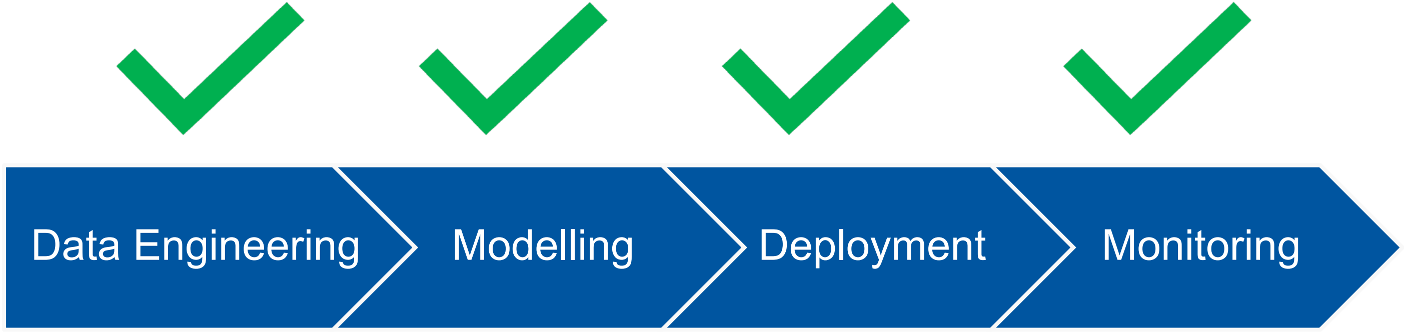
- Model metrics
  - Distribution of input features – data/concept drift
  - Missing/malformed values in the input
  - Average output accuracy/classification distribution – concept drift

- Infrastructure metrics
  - Logging errors
  - Memory, CPU resources utilization
  - Latency and jitter

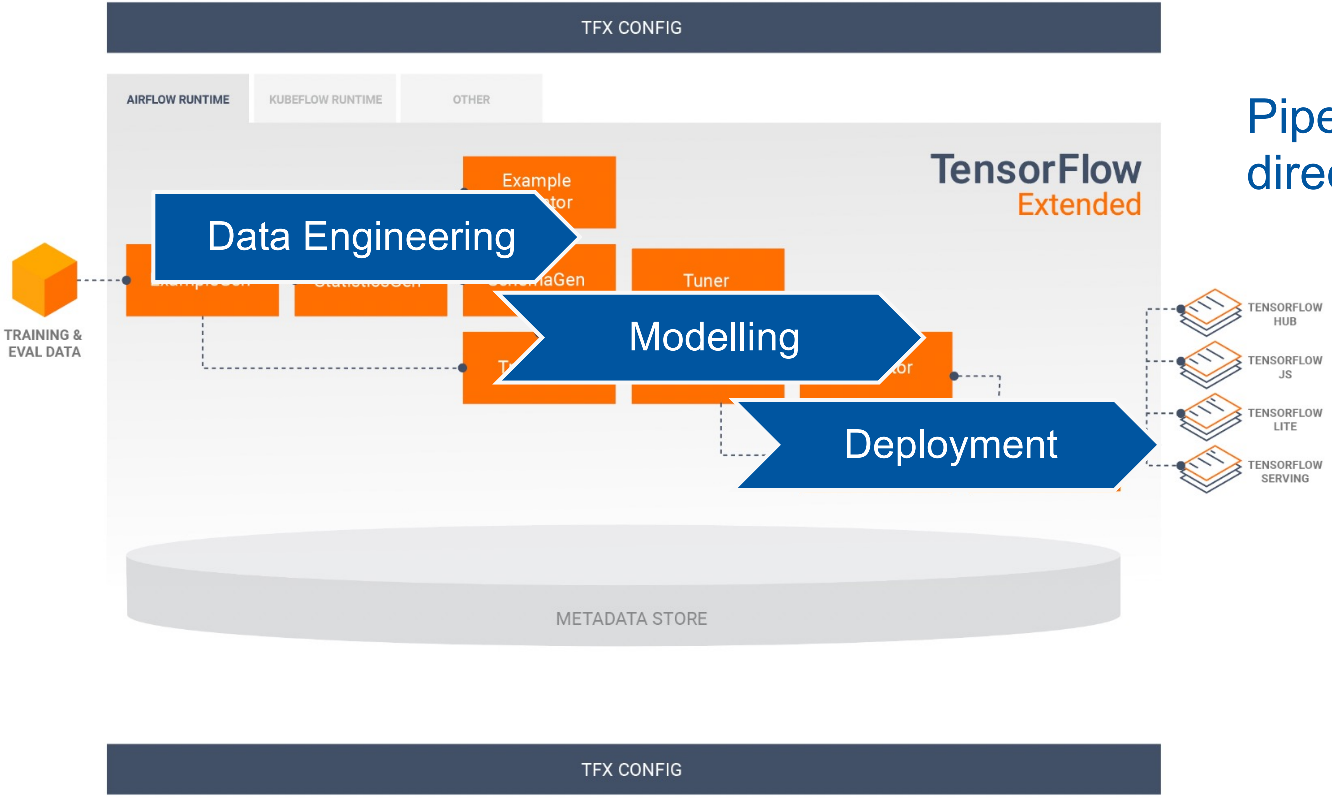
For each of the relevant metrics one should define warning/error thresholds.

Monitoring Matters




Data Engineering
Modelling
Deployment
Monitoring

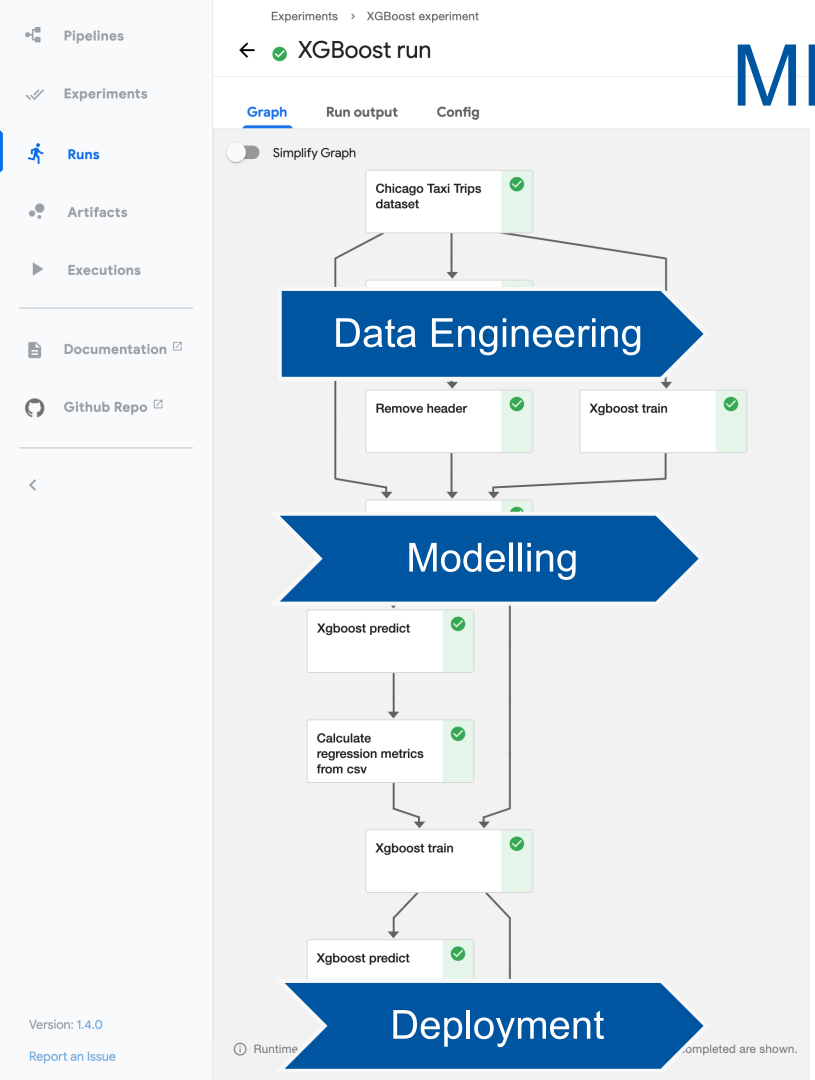
MLOps Pipeline with Tensorflow



Pipeline represented as DAG
directed acyclic graph

https://www.tensorflow.org/tfx/guide

MLOps Pipeline with Kubeflow



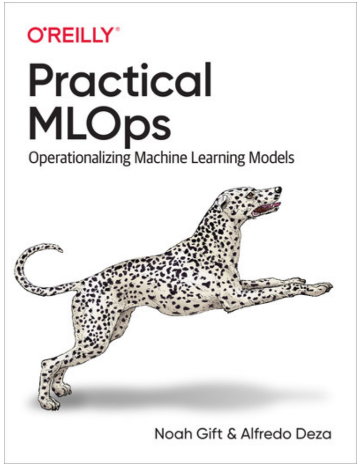
https://ml.cern.ch
https://www.kubeflow.org/docs/started/

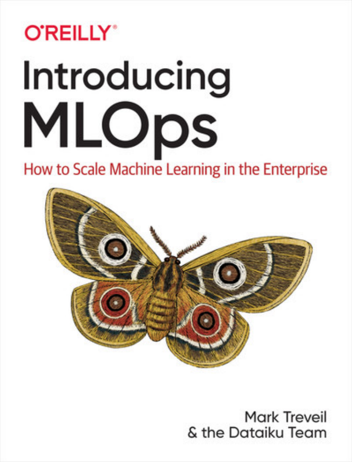
Conclusion

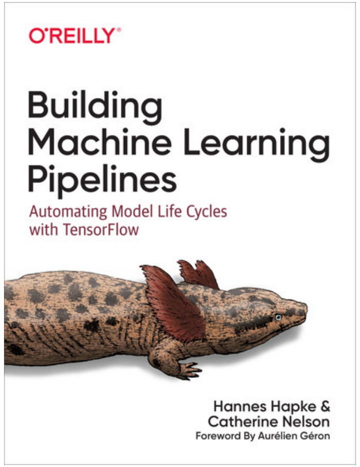
	Development ML	Production ML
Objective	High-accuracy model	Efficiency of the overall system
Dataset	Fixed	Evolving
Code quality	Secondary importance	Critical
Model training	Optimal tuning	Fast turn-arounds
Reproducibility	Secondary importance	Critical
Traceability	Secondary importance	Critical

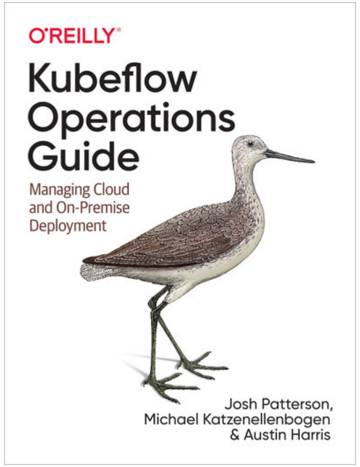
I do hope the presented MLOps concepts will allow your models to transition
From Good to Great

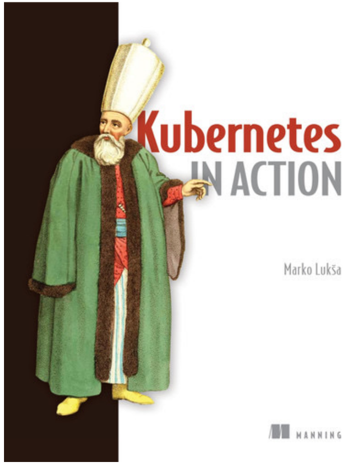
Resources











Machine Learning Engineering for Production (MLOps) Specialization

DeepLearning.AI