From Theory to Practice:

Detecting Model Decay

(or a journey to better understanding of MLOps)

Statistics Finland, Riitta Piela

Our Journey

- At Statistics Finland
  - we had just formed a tiny little machine learning team
  - we had a few ML models running
  - a classification project, which would also build an infrastructure, was just beginning
- We set a goal that we would reach the level 2 of Google Cloud’s MLOps
- MLOps setup includes many components and stages
  - Stages were as Monitoring, Continuous training and Model continuous delivery etc.

MLOps

- "Extension of the DevOps Method to Include Machine Learning and Data Science as First-Class Citizens in DevOps Ecology”
- Why is it so different from DevOps?
- MLOps is the use of the principles and tools of machine learning and traditional software technology in the design and construction of complex systems.
- MLOps covers all steps from data collection to building a model, making the model available and monitoring and maintaining the performance of the model and the entire system.

Train, test and deploy, so simple!?

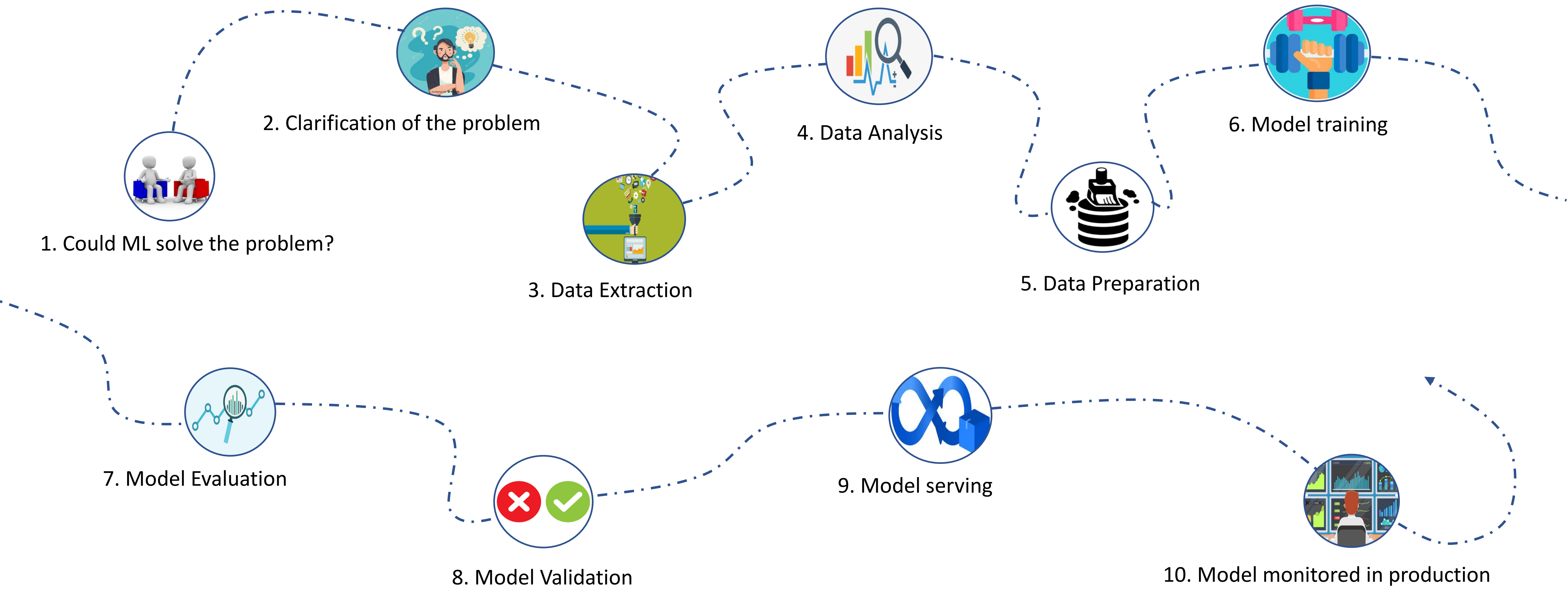
No, one of the biggest mistakes is to assume that models will keep working forever and ever after deployment

A model deployed in production won't be able to adapt to changes in data by itself (we are now talking about offline learning/batch learning)

We understood that model has to be monitored: life in the real world cannot always be predictable

We came to the conclusion that we should have a better understanding of the ML(Ops) process as a whole

Theory: ML(Ops) process



Theory: Data Quality

- We have two quality things, Data Quality and Model Quality, they are separate things and at the same time also connected.
- Data quality monitoring is the first line of defense for machine learning systems. Many issues can be caught before it becomes a model issue.
- Quality in, quality out?

Theory: why degradation happens?

- Usually behind model's degradation is model decay, but what causes model decay?
- Model decay (or model drift or model staleness)
- We started to investigate what those drifts were that caused the model decay
  - we found a large number of different kind of drifts in different articles, but we came to a conclusion that there were really only two different drifts
  - … and If the data quality is fine, two usual suspects behind model decay are data drift and concept drift

Theory: Data Drift

- **Data drift** (feature drift, population drift or covariate shift). Quite a few names to describe essentially the same thing.

  Simplified: the input data has changed. The distribution of the variables is meaningfully different. As a result, the trained model is not relevant for this new data.

  It would still perform well on the data that is similar to the “old” one!

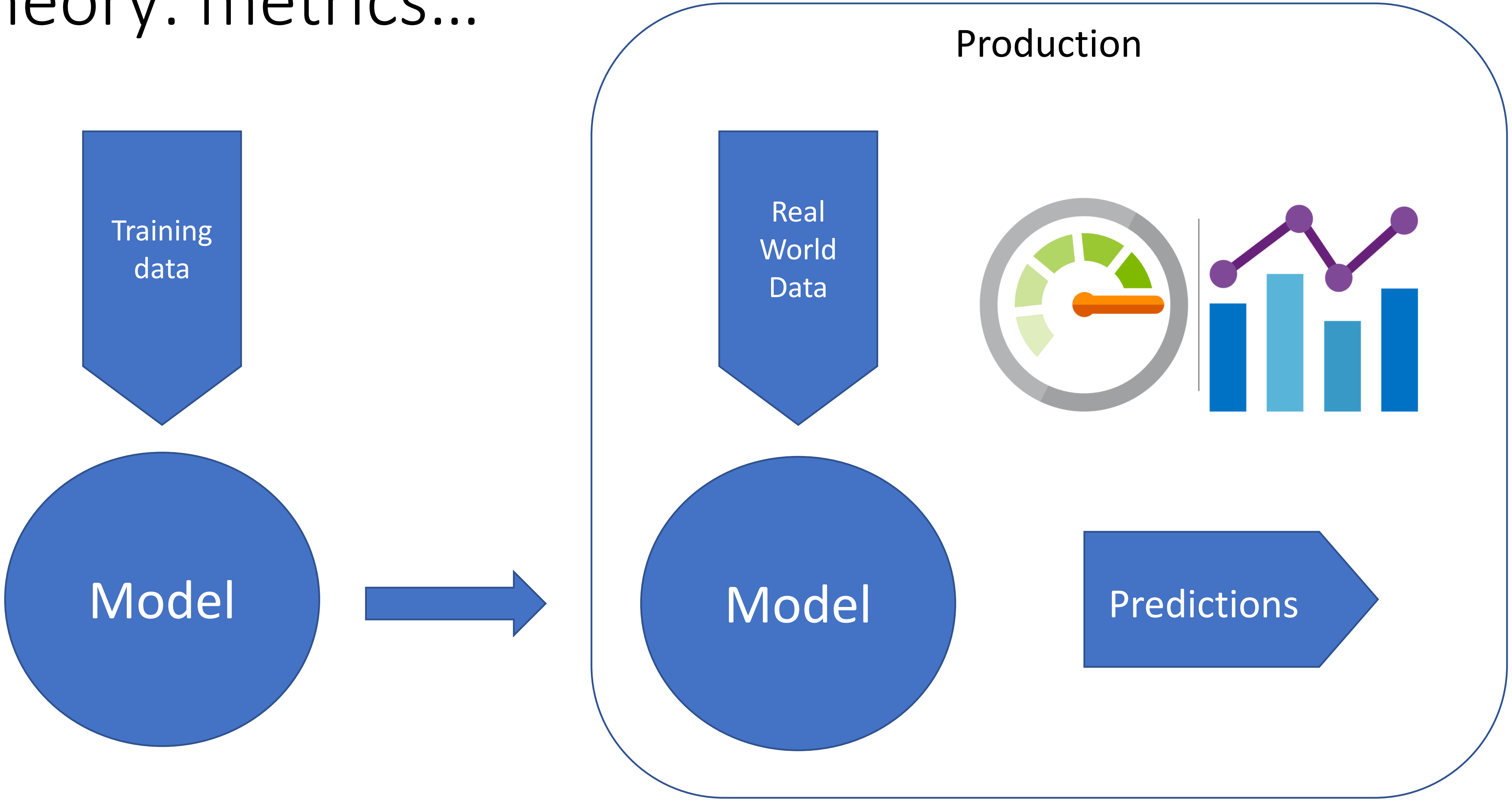
Theory: Concept drift

- **Concept drift** occurs when the patterns the model learned no longer hold.

  In contrast to the data drift, the distributions (such as user demographics, frequency of words, etc.) might even remain the same. Instead, the relationships between the model inputs and outputs change.

  In essence, the very meaning of what we are trying to predict evolves. Depending on the scale, this will make the model less accurate or even obsolete.

Theory: metrics…



Where were we after our theory part?

We had understood

- what is behind model decay
- we had identified other reasons that might cause degradation: training-serving skew, seasonality (”recurring drift”)

We also understood that

- these ”rules” we had identified, they apply to so called batch learning, stream learning is little different…
- data quality is really crucial, but you never know how the real world changes
- Metrica and methods, there were just so many of them…

Practice: detecting Data Drift

- Data Drift (change in the distribution of variables or predictors)
- 1. We trained the model in production
- 2. We manipulated data so that data drift occurs: simulating data drift
- 3. Methods we used to detect data drift
  - We started with Alibi Detect –libraries Kolmogorov-Smirnov algorithm
  - We also used Evidently

Links: In GitHub and Drift Detection analysis in Yupiter Notebook

Practice: Detecting Concept Drift

- Concept Drift was: a shift in the actual relationships between model inputs and output. P(Y|X). In contrast to the data drift, the distributions might even remain the same.
- Little bitt trickier to simulate because concept drift comes in different flavors (many types)
- Libraries and algorithms
  - We started with the same libraries as we used with data drift

How to decide when to retrain?

- It can be based on many different factors
  - Retraining based on interval
  - It can be triggered on performance-base
  - It can be triggered based on data changes
  - Retraining just on demand
- But, the best solution is to monitor and automate the retraining on performance base

Retraining approaches vary

- Retrain the model using all available data, both before and after the change
- Use everything, but assign higher weights to the new data so that model gives priority to the recent patterns
- If enough new data is collected, we can simply drop the past data
- But it would be good to have a retraining strategy (for different types of learning)

Monitoring tools and libraries

- Hydrosphere
- Amazon SageMaker Model Monitor
- Fiddler
- Alibi Detect library
- Evidently

Lessons learned

- The larger scope is model care and maintenance
- Much of this care and maintenance should be automated
- MLOps takes care of this automation
- MLOps architecture includes components and processes that are absolutely necessary
- This journey started from studying MLOps and now at the end of the journey MLOps seems even more important

Thank you!

- This journey started from studying MLOps and now at the end of the journey MLOps seems even more important