Maximizing the Impact of ML Systems Through Software Engineering Practices

A Research-Centric Approach to Development, Deployment, and Maintenance

Karlstad 31 October 2023

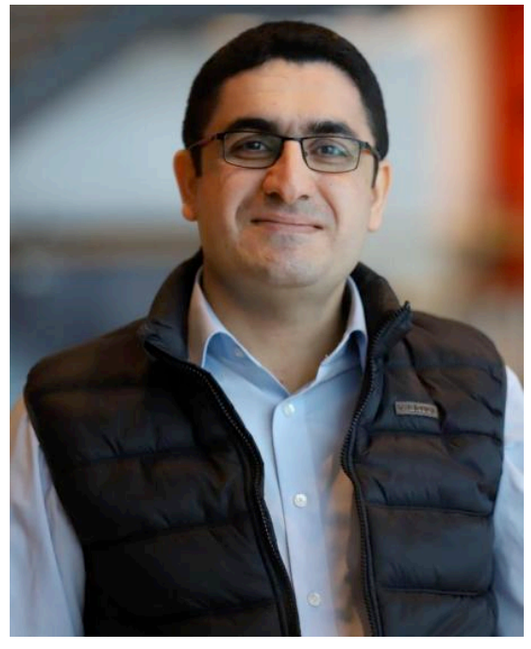
Bestoun S. Ahmed
Professor in Computer Science
Department of Mathematics and Computer Science
Karlstad University, Sweden
Phone: +46 54 700 1861
Room: 21D 413
bestoun@kau.se

DAMI4.0

KARLSTADS UNIVERSITET

About Me

- Bestoun S. Ahmed, Ph.D.
- Professor in Software Engineering
- 15 Years of Software Engineering and AI Research
- More information about me, please visit www.bestoun.net
- Email: bestoun@kau.se
- Phone: +46-54 700 1861
- Room: 21D 413



- Office Address:
- Department of Mathematics and Computer Science,
- Faculty of Health, Science and Technology,
- Karlstad Univesity, Sweden

The Current Landscape of SE and AI

Software 1.0

- The “classical stack” of **Software 1.0** is what we’re all familiar with.
- The program is written in languages such as Python, C++, etc.
- It consists of explicit instructions to the computer written by a programmer.
- By writing each line of code, the programmer identifies a specific point in program space with some desirable behaviour.
- This process of an explicit description of problem solving steps is what is used to be **Software 1.0**

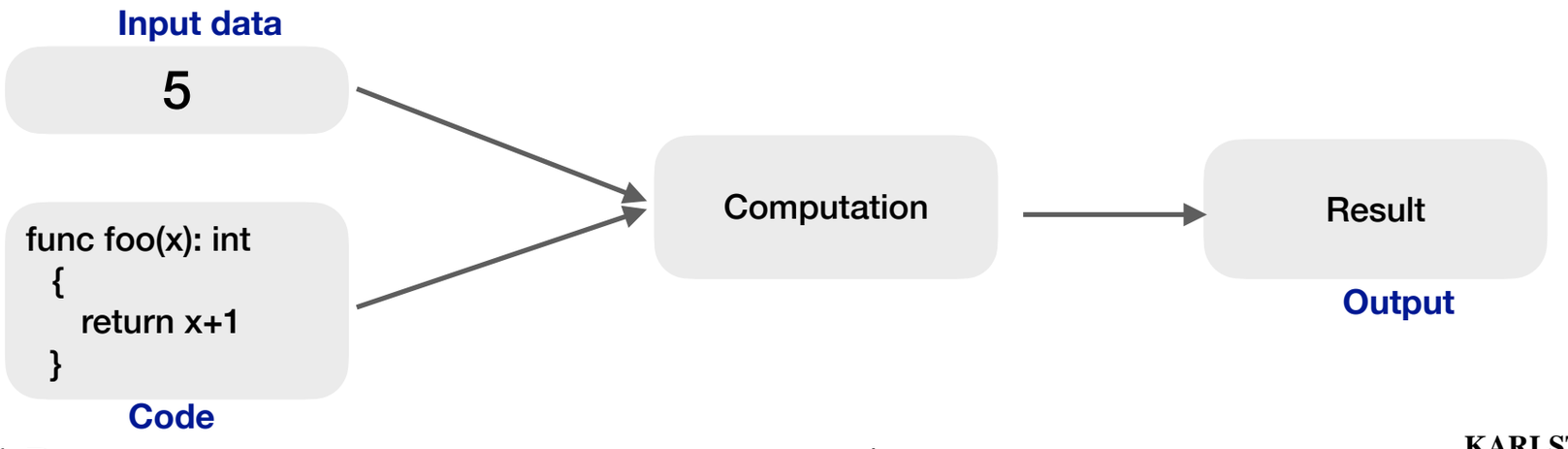
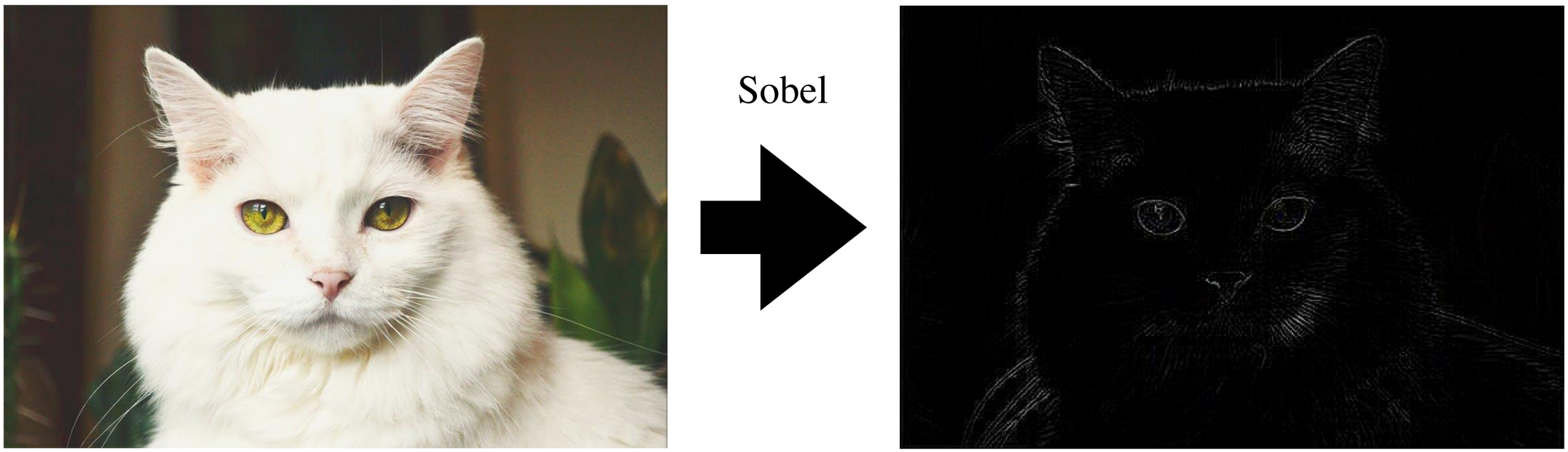


Diagram source: https://karimfanous.substack.com/p/software-20-vs-software-10?utm_source=url

Software 1.0

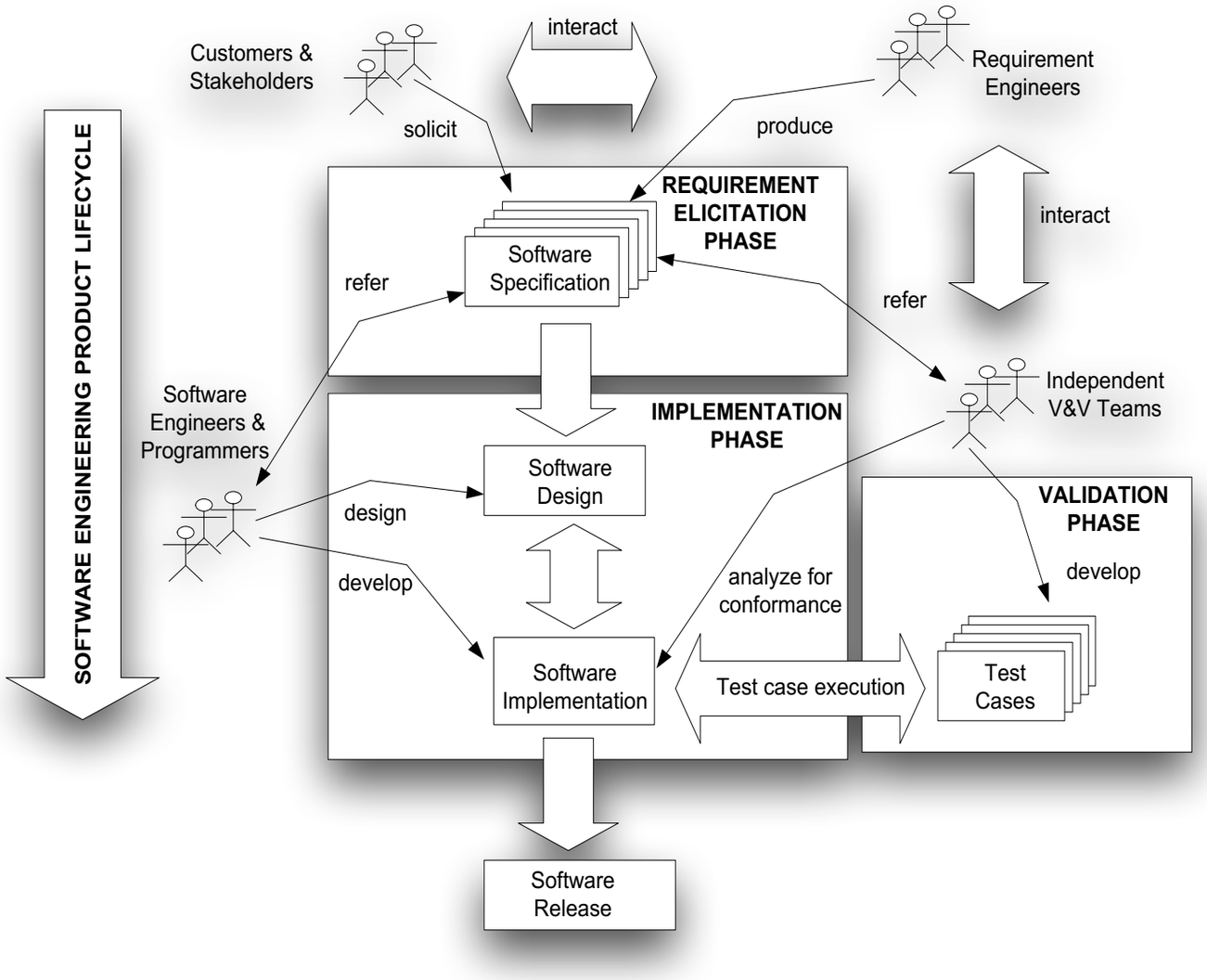
- Great for problems that are well defined, e.g.:
  - TCP/IP Stack: layered stack of protocols to facilitate reliable data transfer over the internet
  - Computer Graphics: geometric shapes to 2D pixels on a screen
- Cannot deal properly with complex tasks, e.g., image recognition



Source: https://www.lwtech.edu/

Top view of software development activities

- Decompose large problems into smaller, easier problems to solve



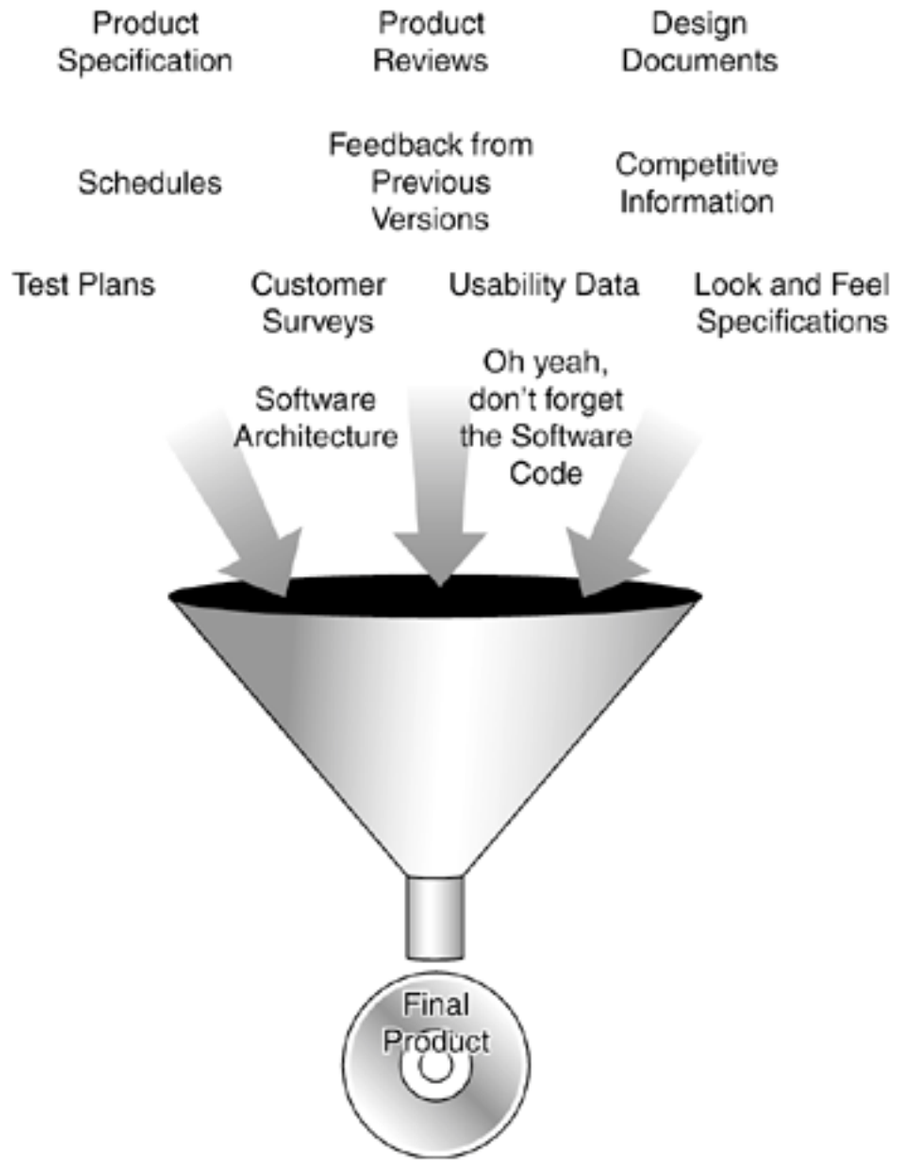
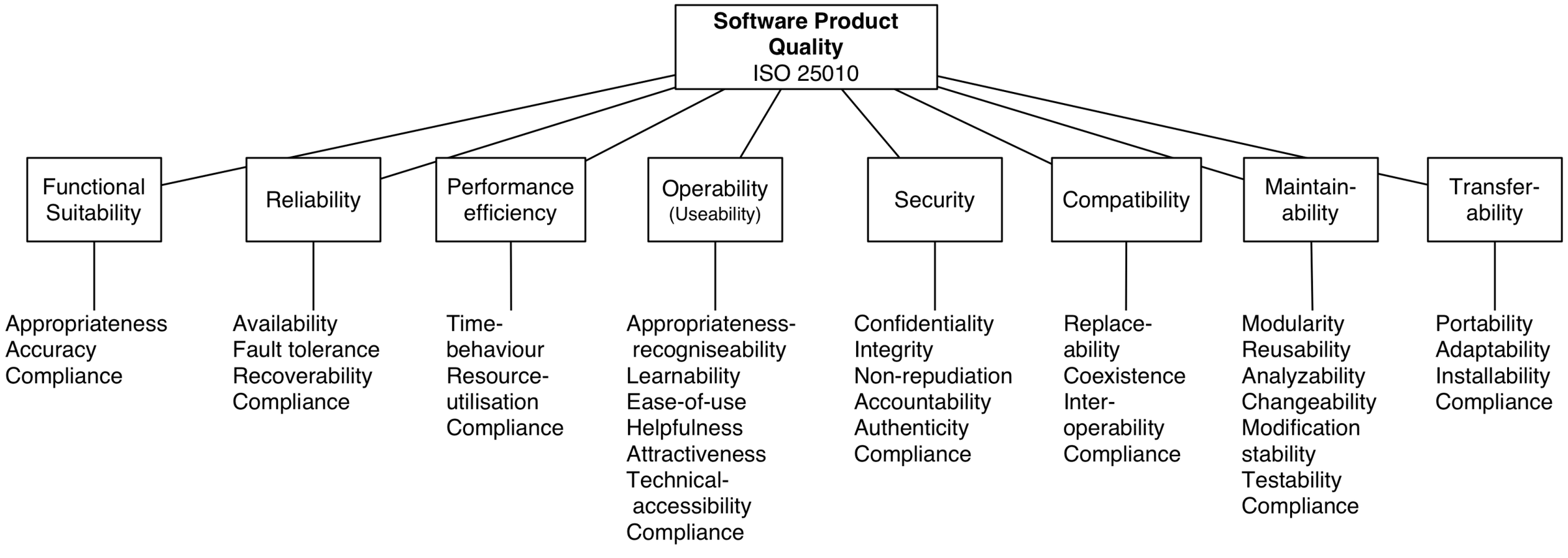


Figure Source: The Art of Software Testing (3 edition)

- A lot of hidden effort goes into a software product.

Modern Software Engineering Activities

- Build a product
- Concerned about cost, performance, stability, schedule
- Identify quality through customer satisfaction
- Scale the solution and detect errors, consider security, safety, fairness .. etc.
- **Maintain and evolve the product over long period.**



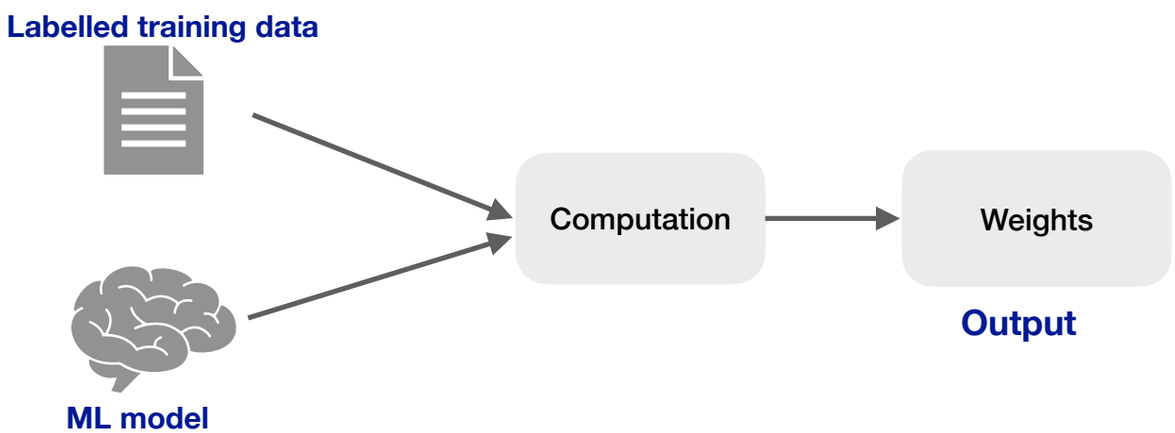
Software 2.0

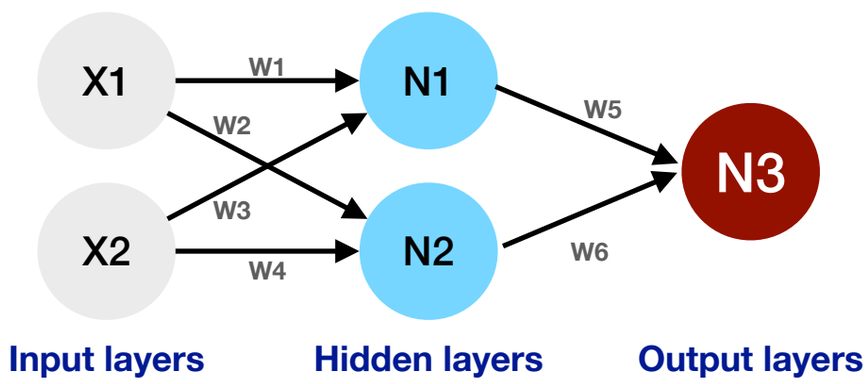
- **Software 2.0** is about finding programs through optimisation i.e. directed search using training data as the guide.

- **Software 2.0** can be written in much more abstract, human unfriendly language, such as the weights of a neural network.

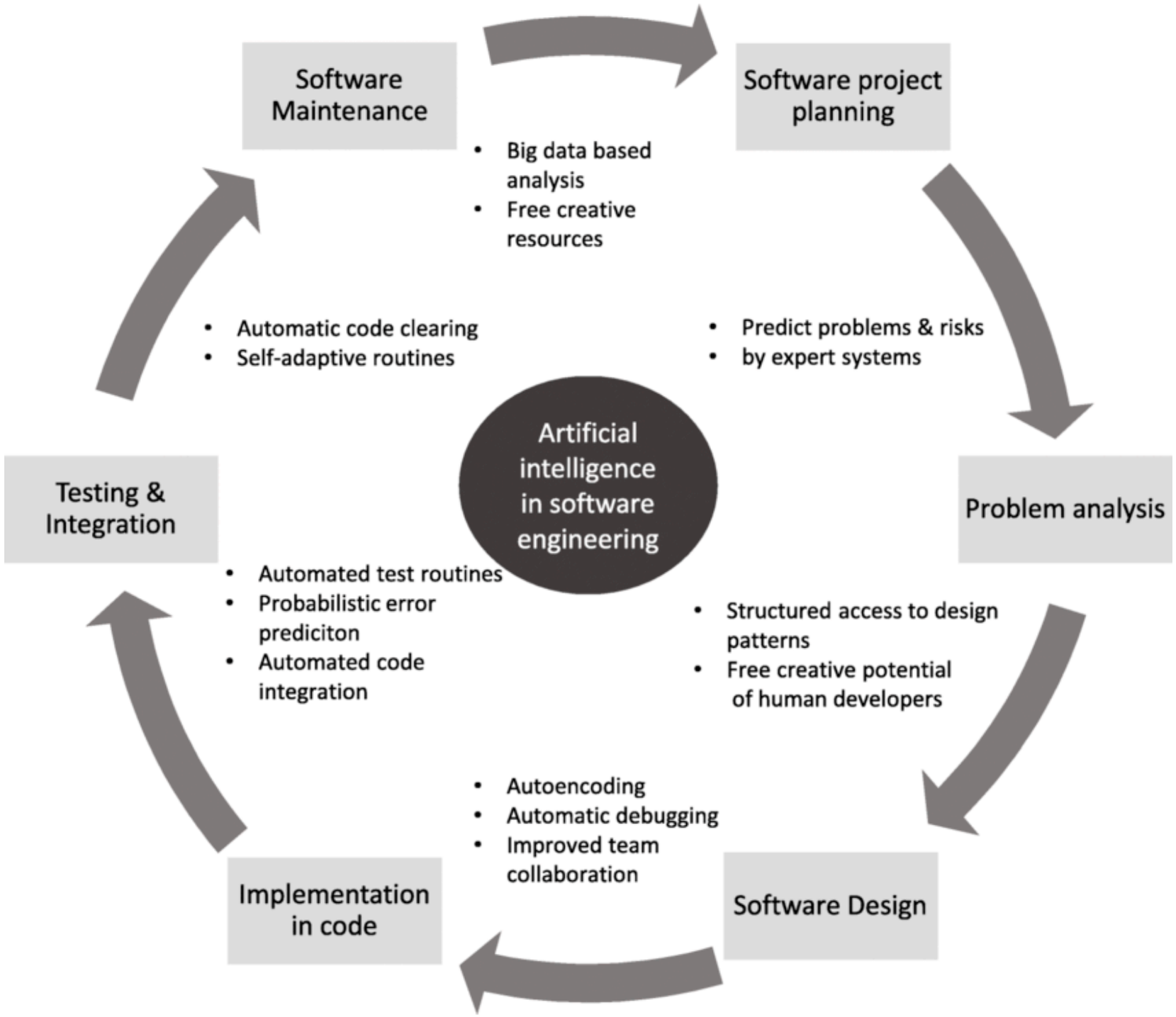
- No human is involved in writing this code because there are a lot of weights (typical networks might have millions), and coding directly in weights is kind of hard.

- We used to create all the logic in our programs, but now because we're using gradients and deep learning and automation, we can start tackling more complex tasks.



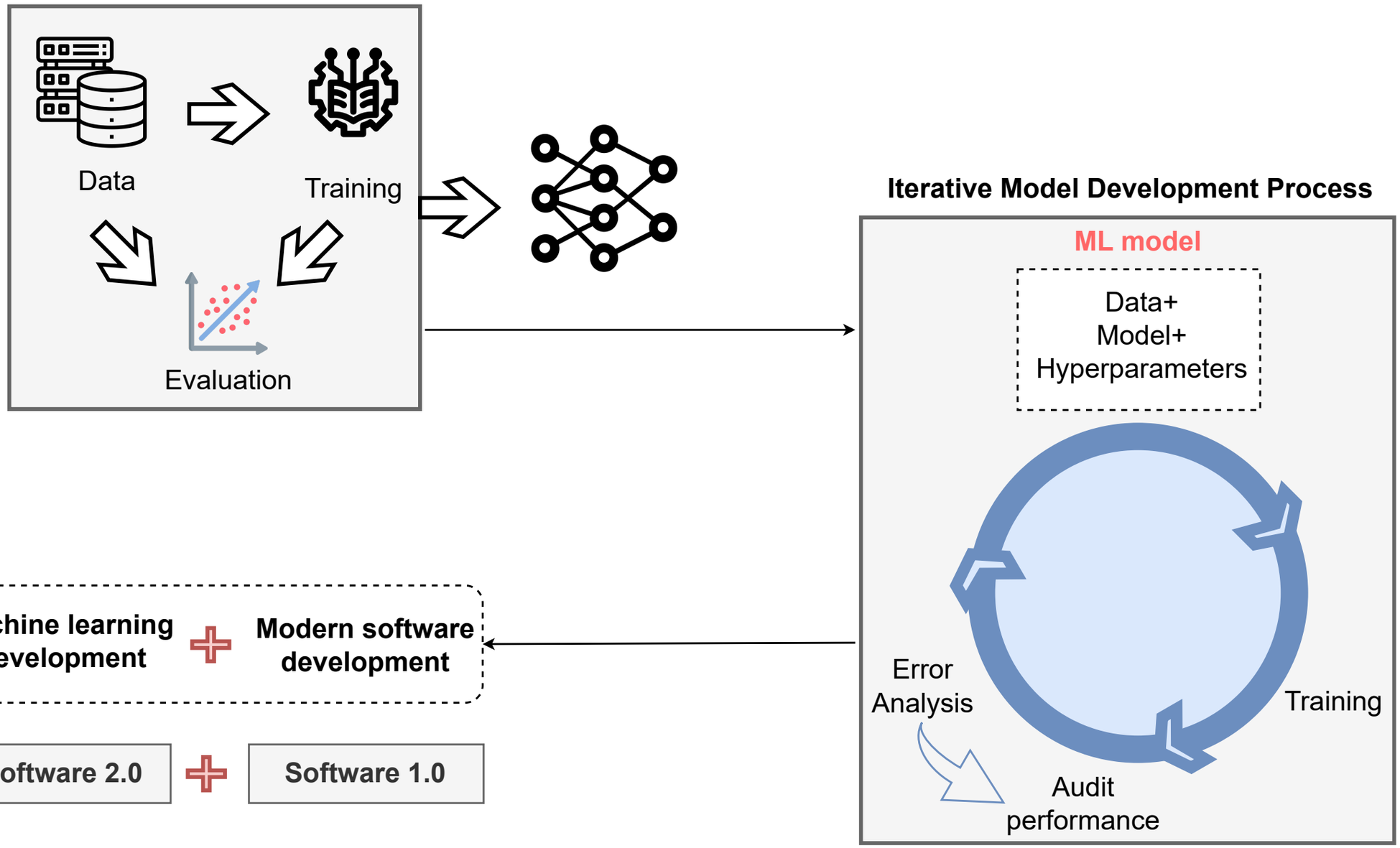


AI Development life cycle

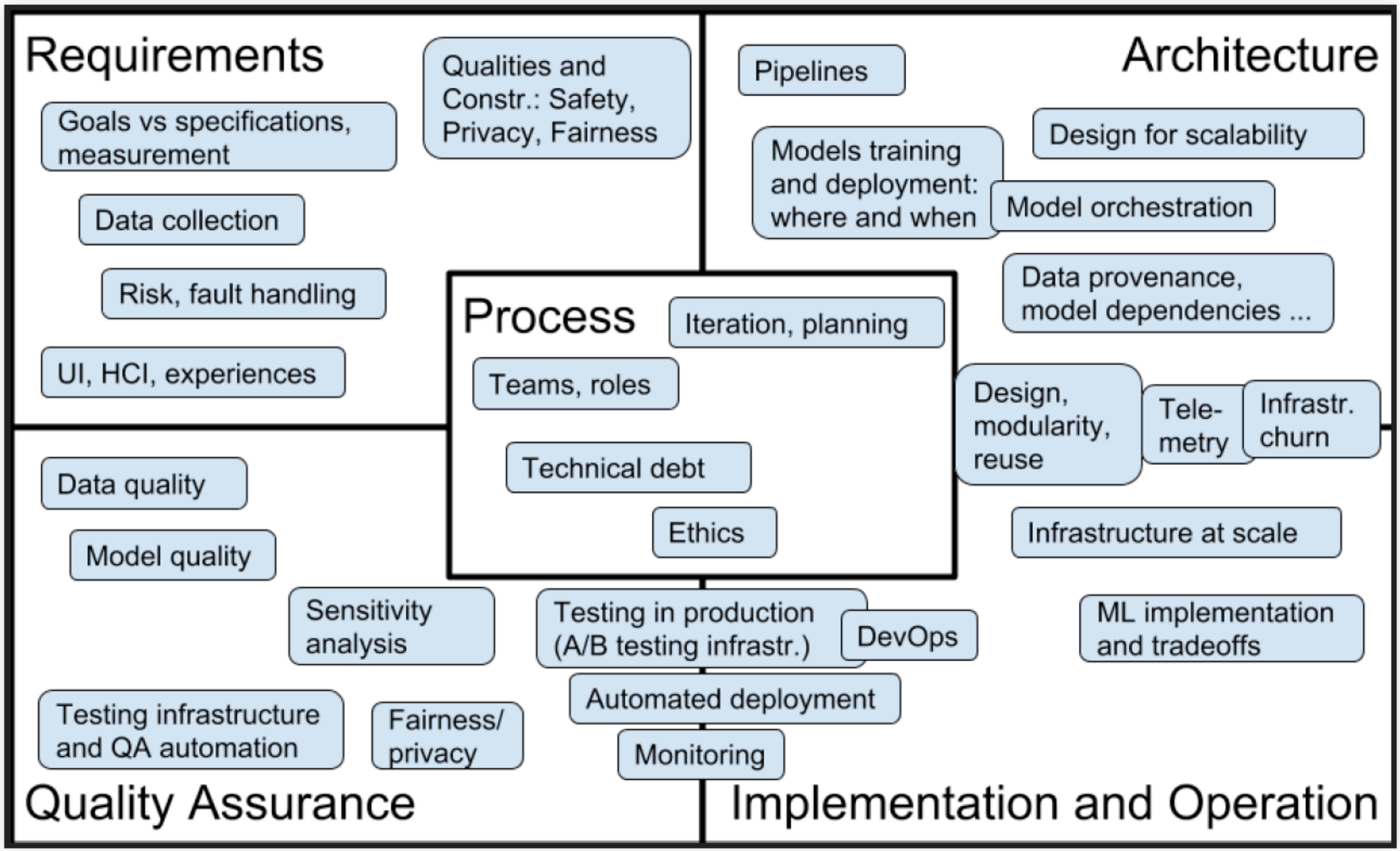


Source:Applications of AI in classical software engineering

Moving Forward from Traditional ML Modelling



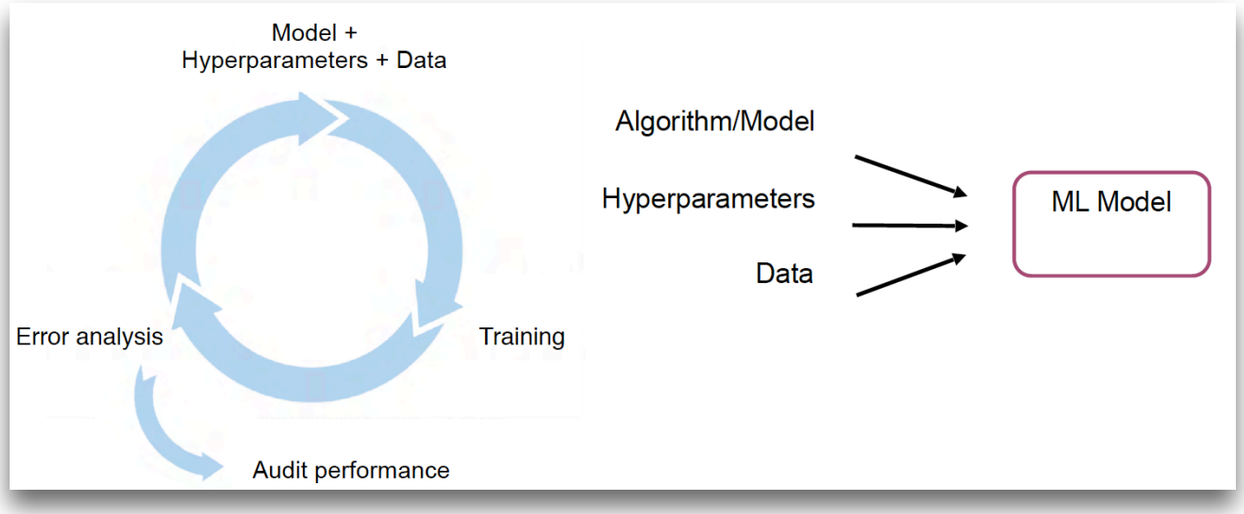
Inline with SE Dimensions

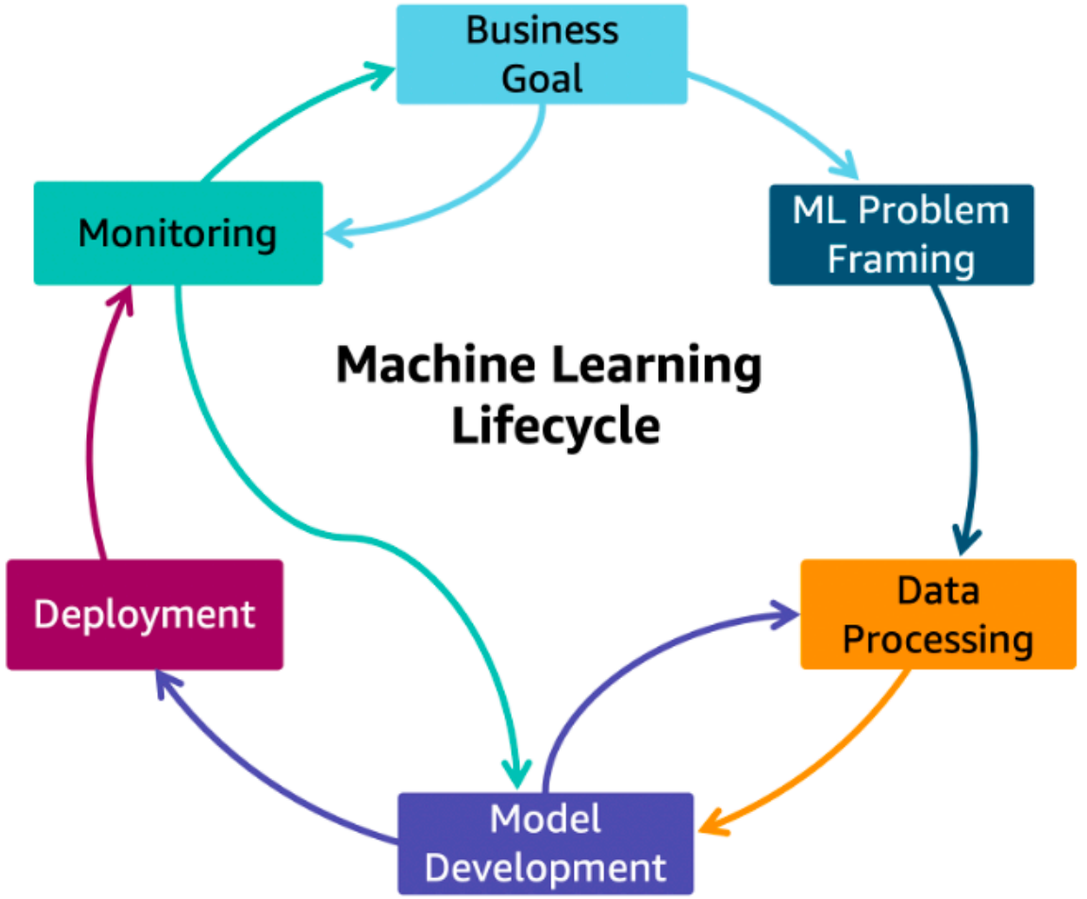


Christian Kästner and Eunsuk Kang. 2020. Teaching software engineering for AI-enabled systems(ICSE-SEET '20).

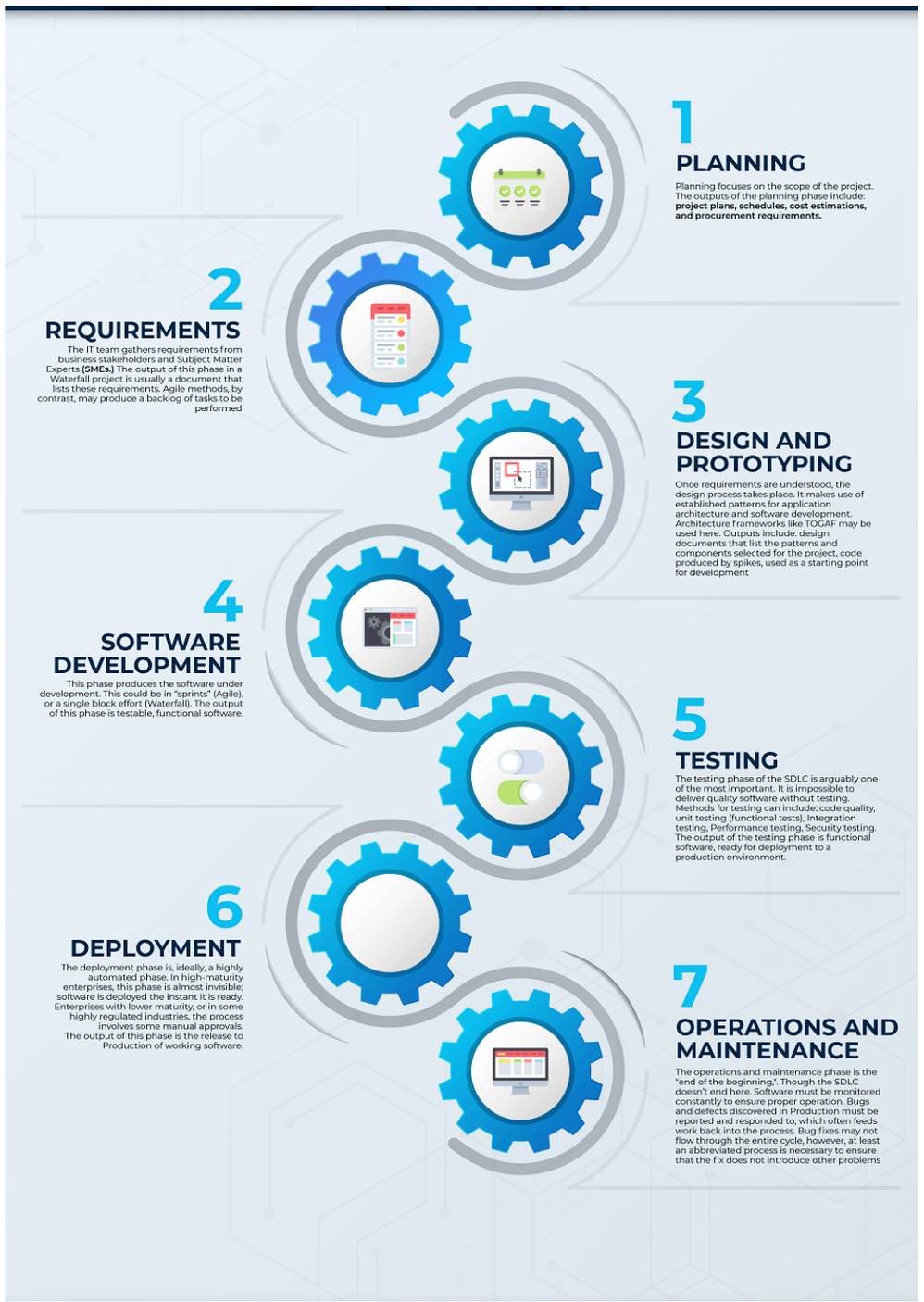
ML Development Lifecycle

Classical iterative approach





Synergy with Software Eng.

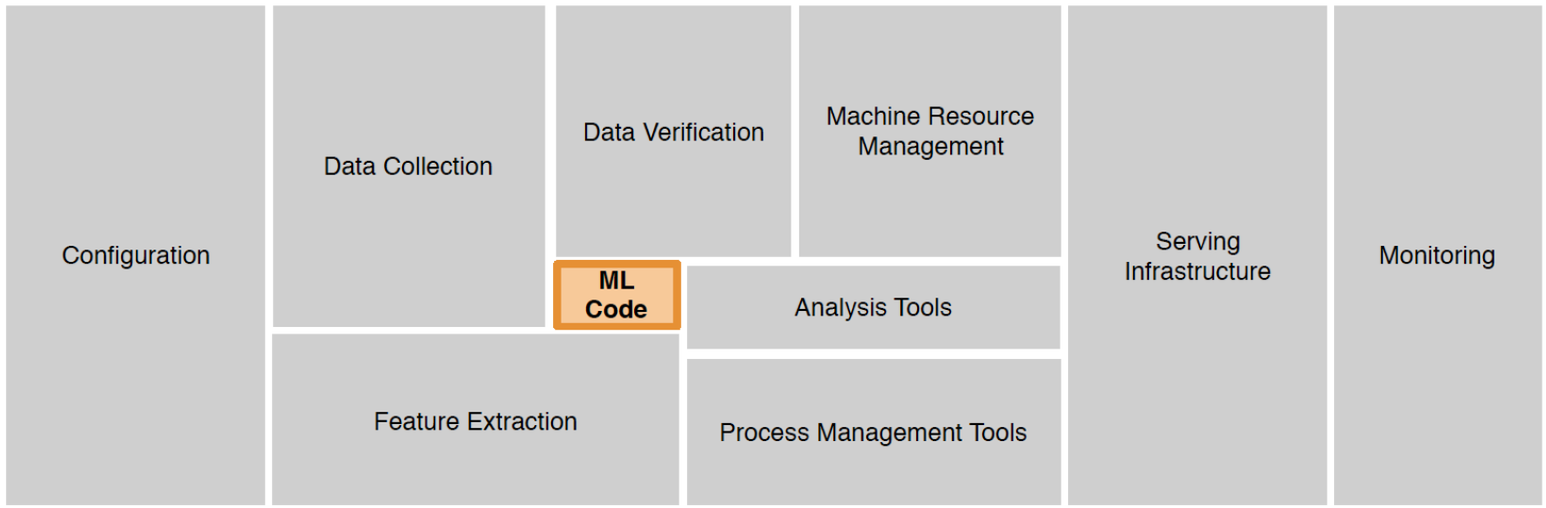


Figures source: https://raygun.com/ and https://docs.aws.amazon.com

Why Software Engineering for AI

- Machine learning components are parts of much larger systems

- Many existing SE practices apply directly but are simply not used in the data science field



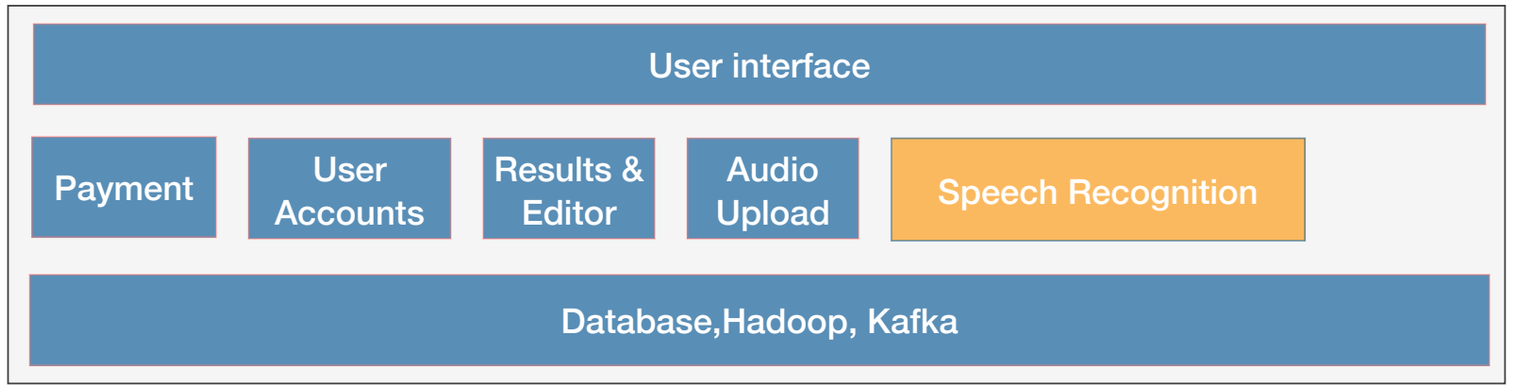
- ML-enabled systems need to be engineered such that
  - System is instrumented for runtime monitoring of ML components and operational data
  - Training-retraining cycle is shortened
  - ML component integration is straightforward

- Other SE practices will have no be adapted to extended to deal with ML components

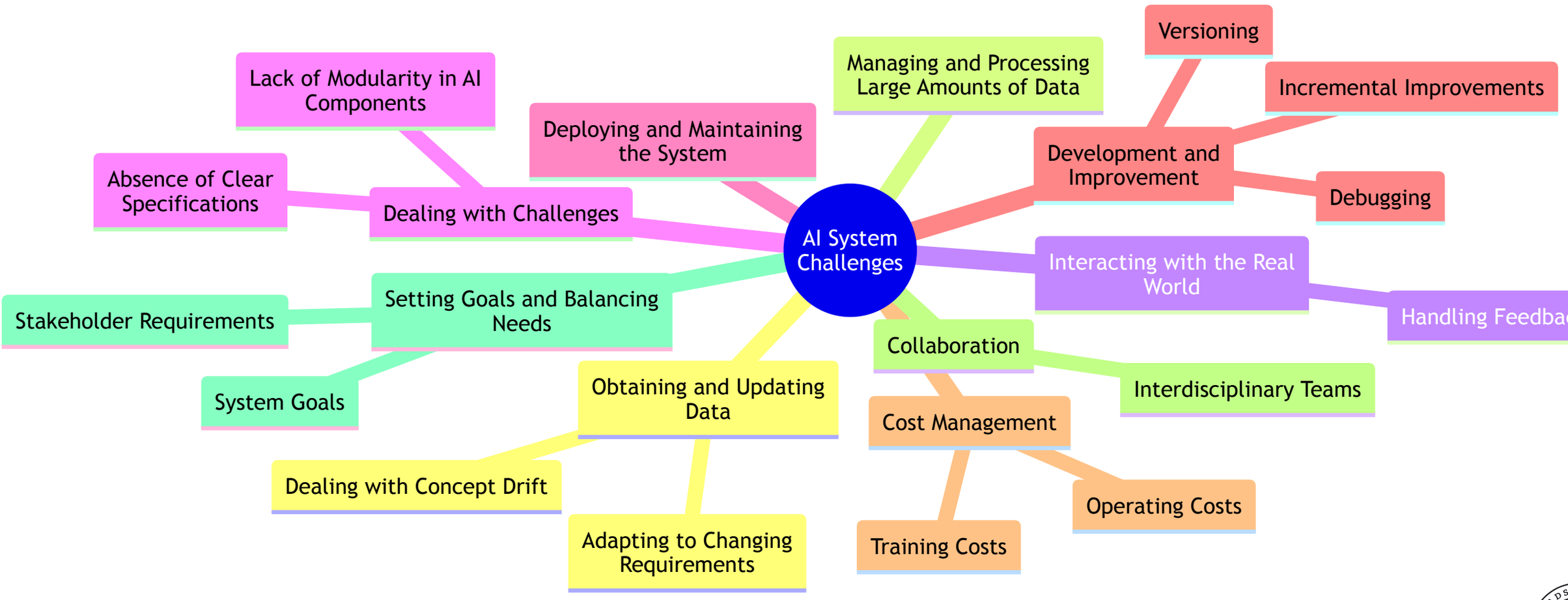
Source: [D. Sculley et. al. NIPS 2015: Hidden Technical Debt in Machine Learning Systems]

Integrating ML as a Core Component in Complex Systems

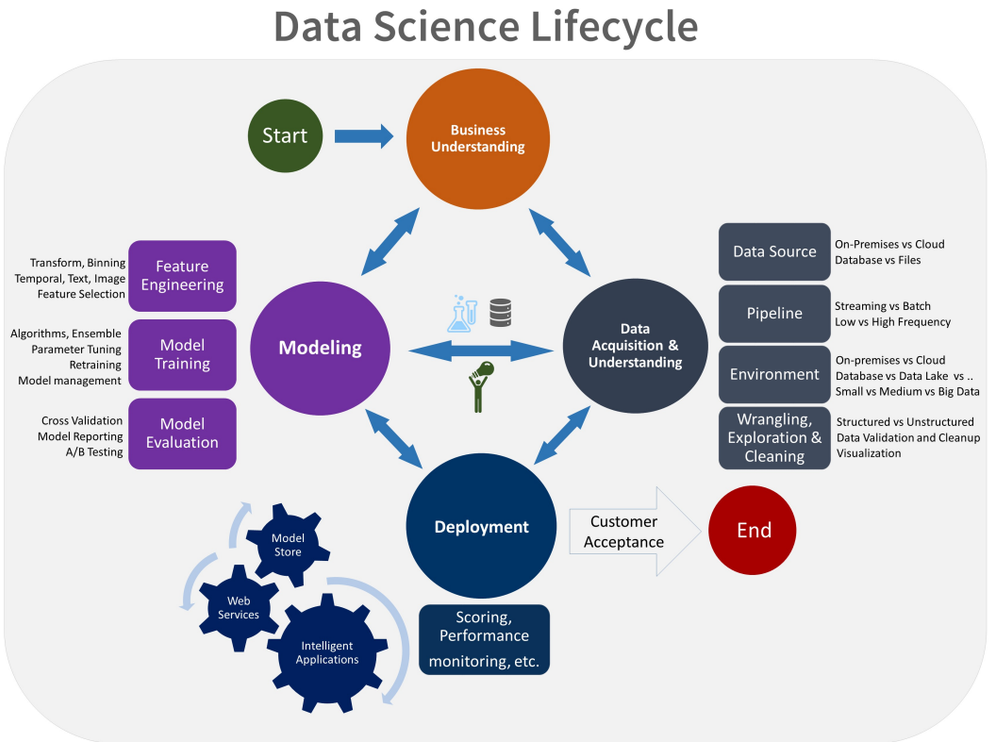
- ML algorithm is just a smal part of a bigger system

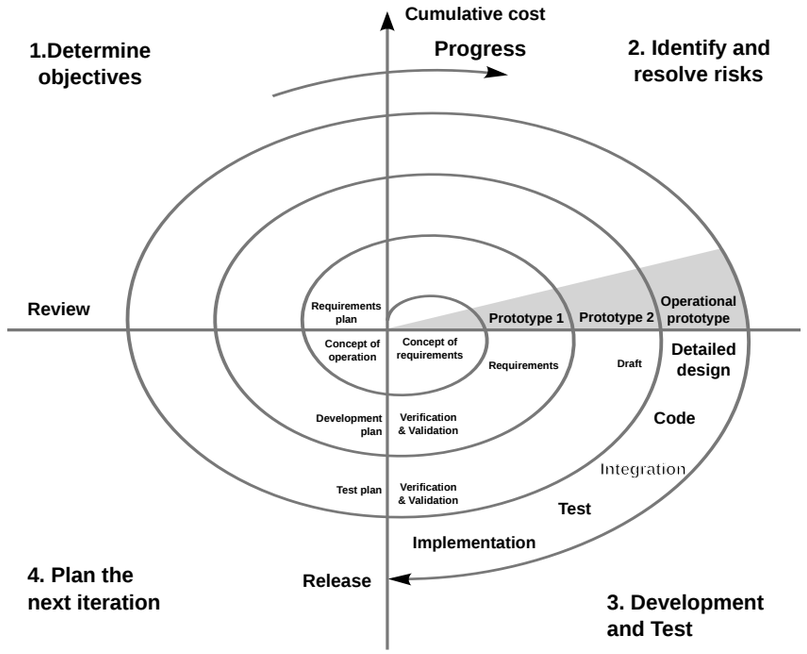


- **There are several other challenges:**



Data Science and Software Engineering





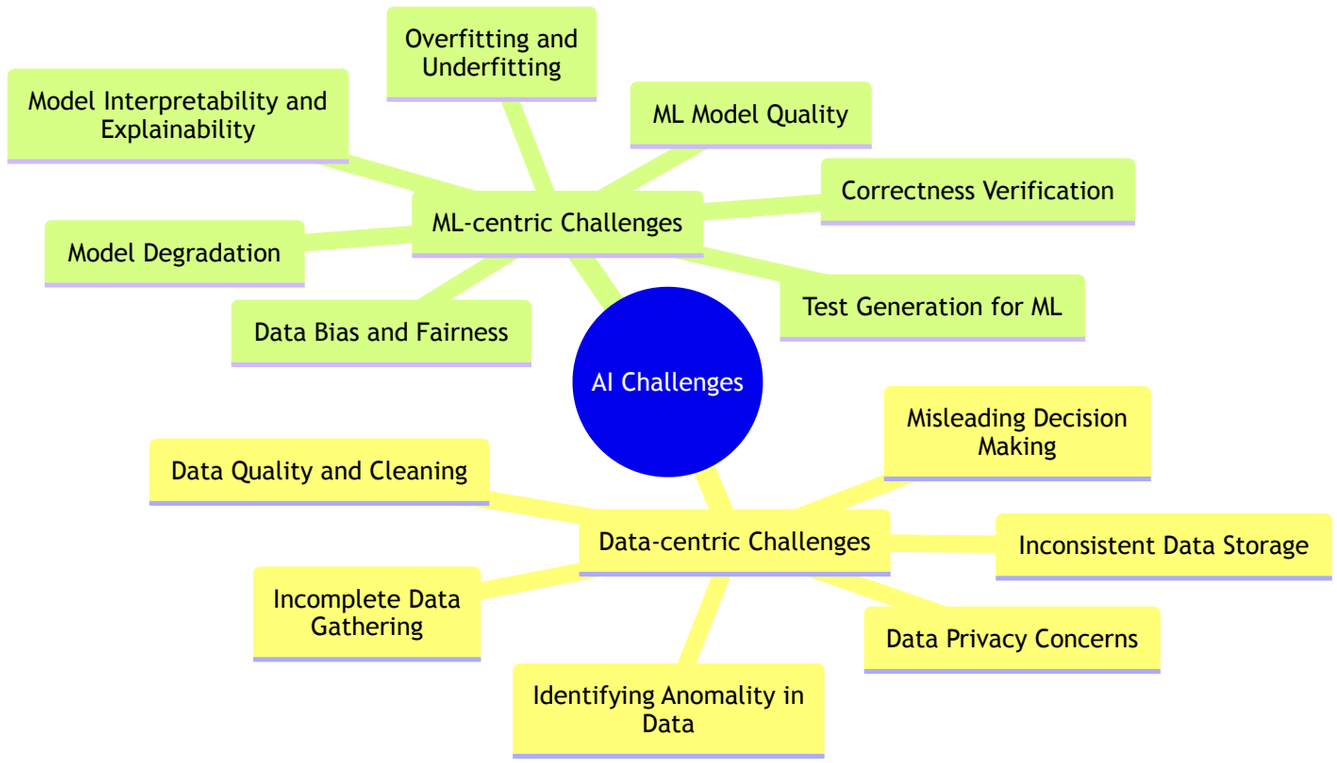
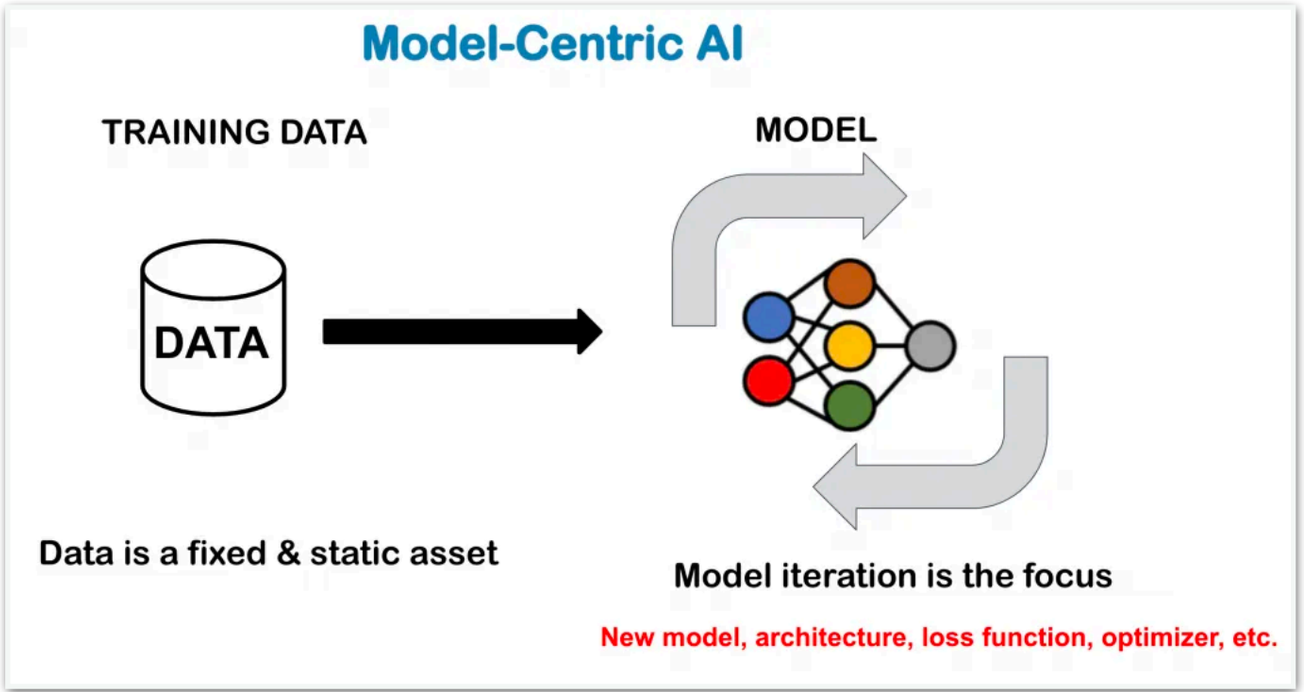


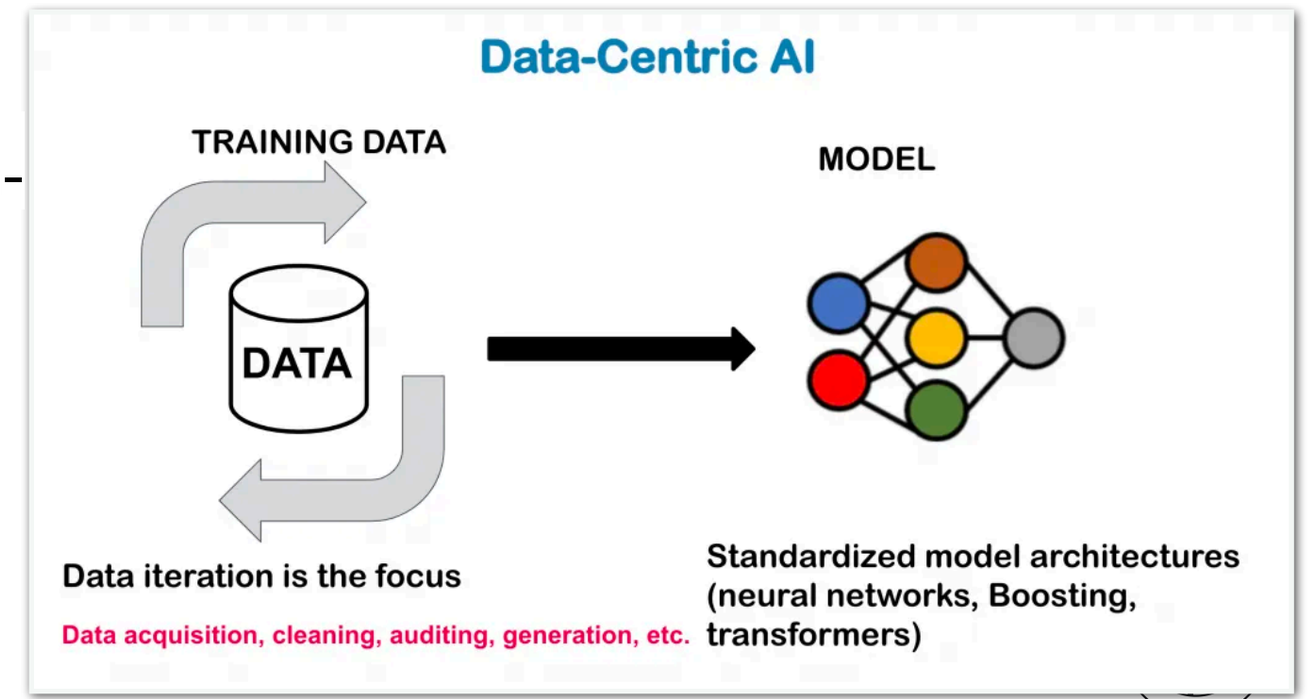
Image Source: (Microso Azure Team, "What is the Team Data Science Process?" Microso Documentation, Jan 2020)

AI = Data + Model

- **Model-Centric AI:**
  - Emphasis on developing and refining complex algorithms and models.
  - Focus on optimizing model performance and accuracy.
  - Prioritizes the design and architecture of the model.



- **Data-Centric AI:**
  - Emphasis on acquiring and curating high-quality data.
  - Focus on data preprocessing, cleaning, and augmentation.
  - Prioritizes data quality, diversity, and relevance.



nderschaar-lab.com/dc-check/what-is-data-centric-ai/

General Data-centric problems with ML

Data Quality Issues

- **Types of data quality issues:**
  - **Completeness:** Missing or incomplete data.
  - **Consistency:** Conflicting or duplicate data.To what extent the data are compatible with the previous values or rules that apply to the data.
  - **Accuracy:** Incorrect or outdated data. The level to which the data is correct, reliable, and certified.
  - **Timeliness:** Delayed or stale data. For some tasks, the data might become irrelevant very quickly.
  - **Relevancy:** The requirement for the data to be sufficient to achieve the business goals.

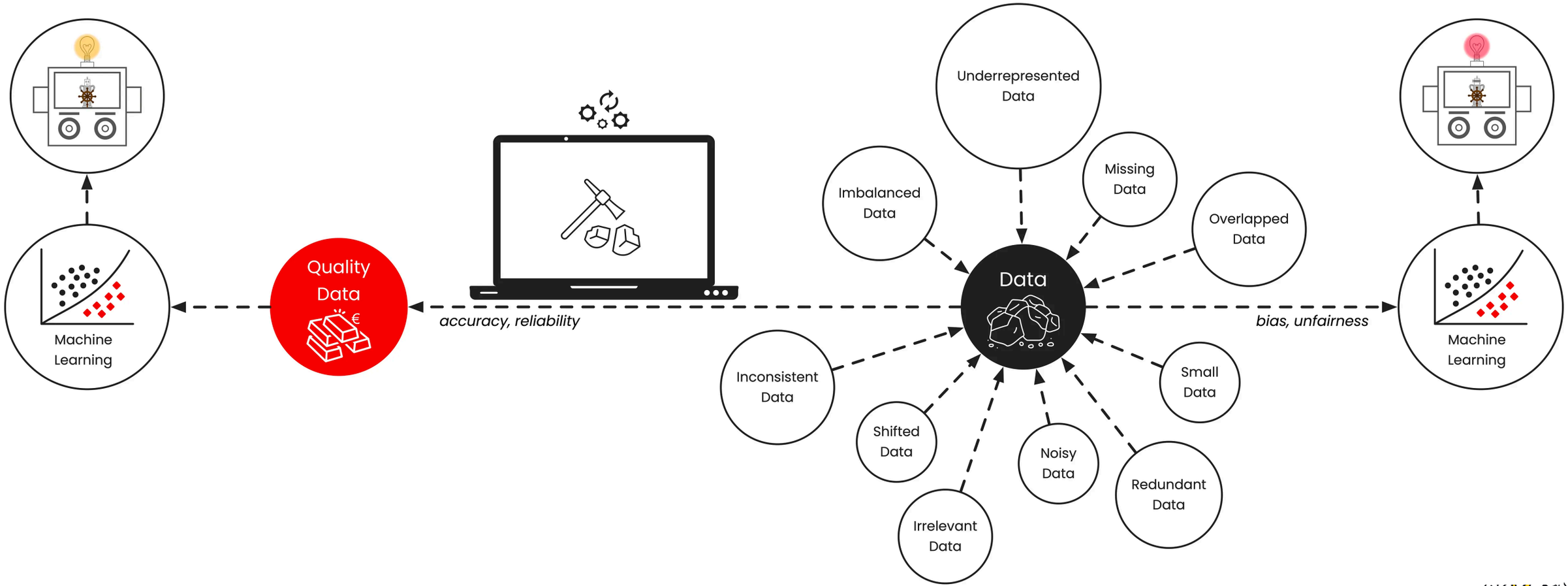
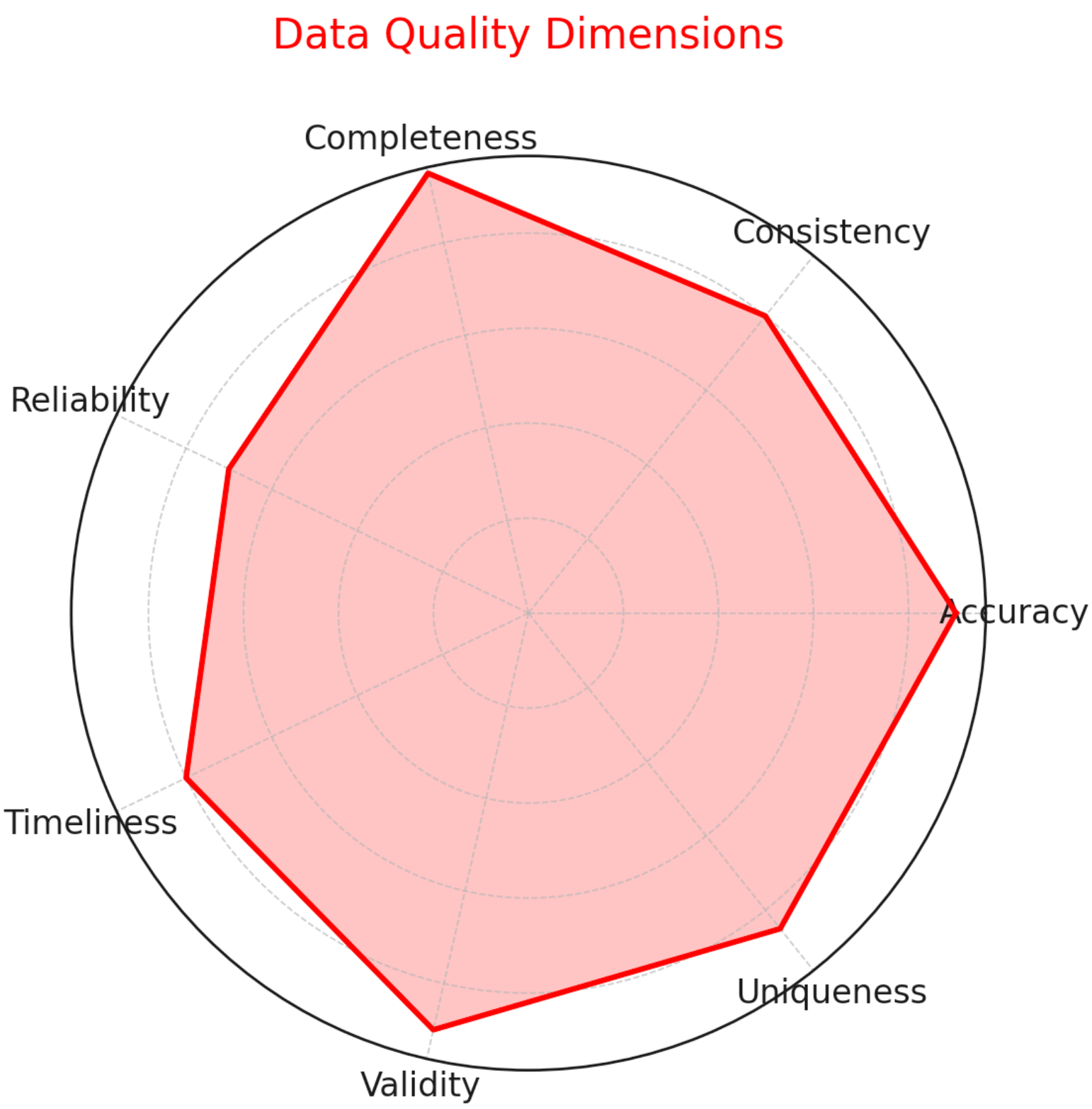
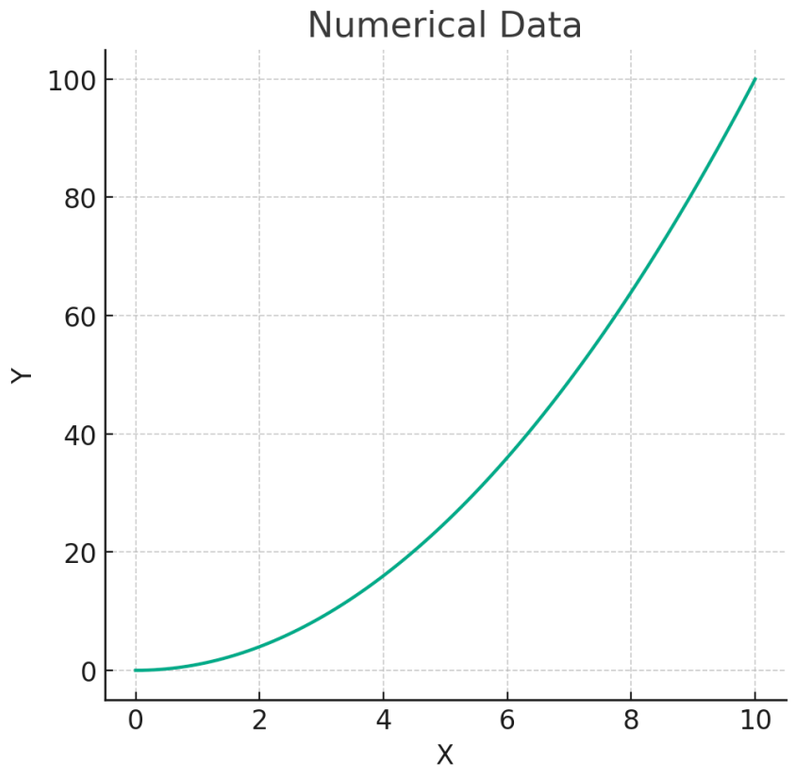


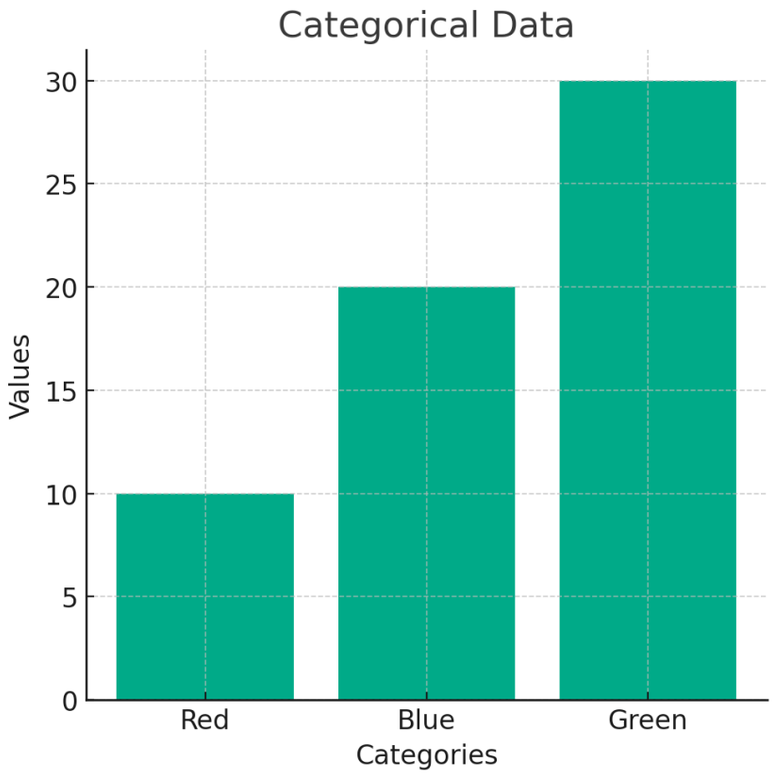
Image source: https://towardsdatascience.com/data-quality-issues-that-kill-your-machine-learning-models-961591340b40

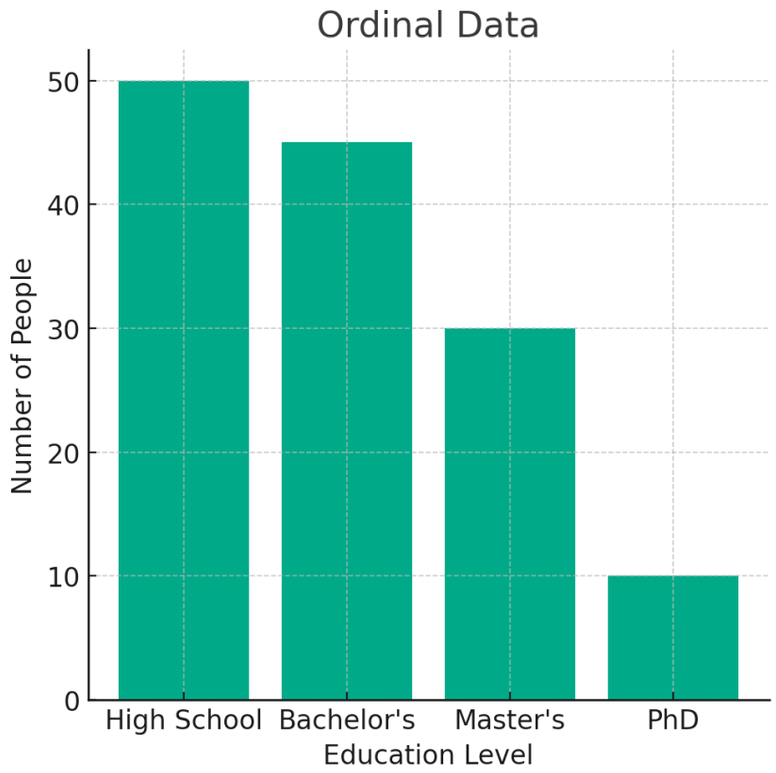
Data quality dimensions

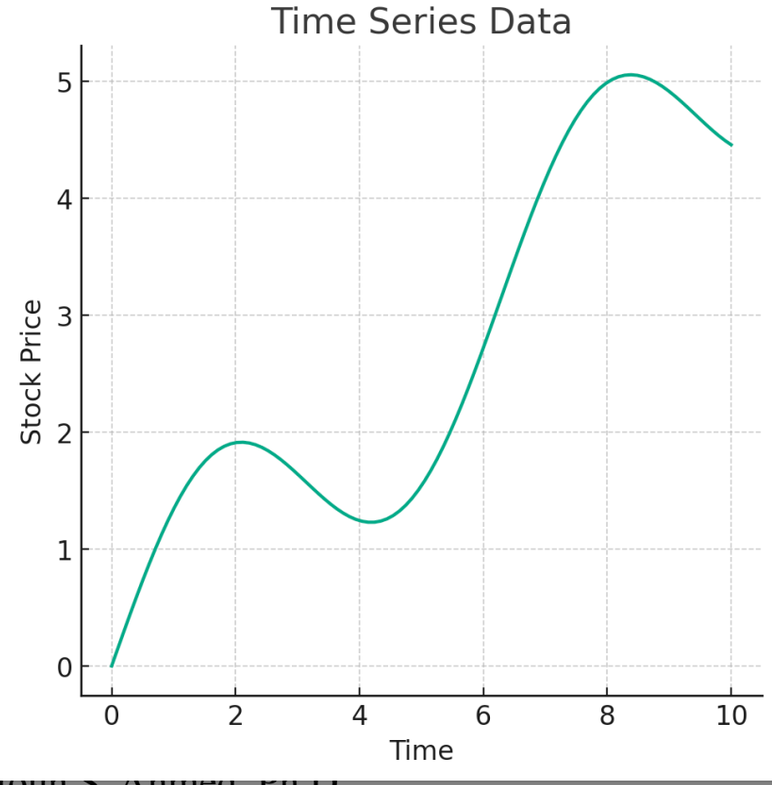


Data Types



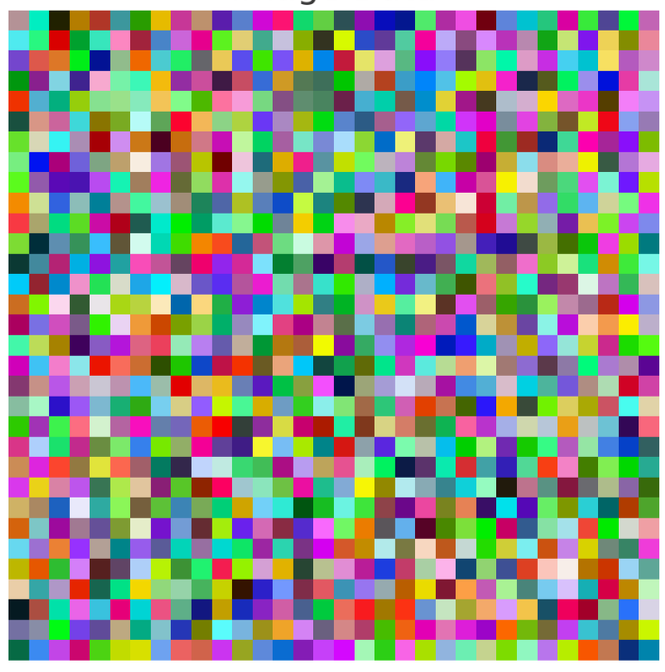






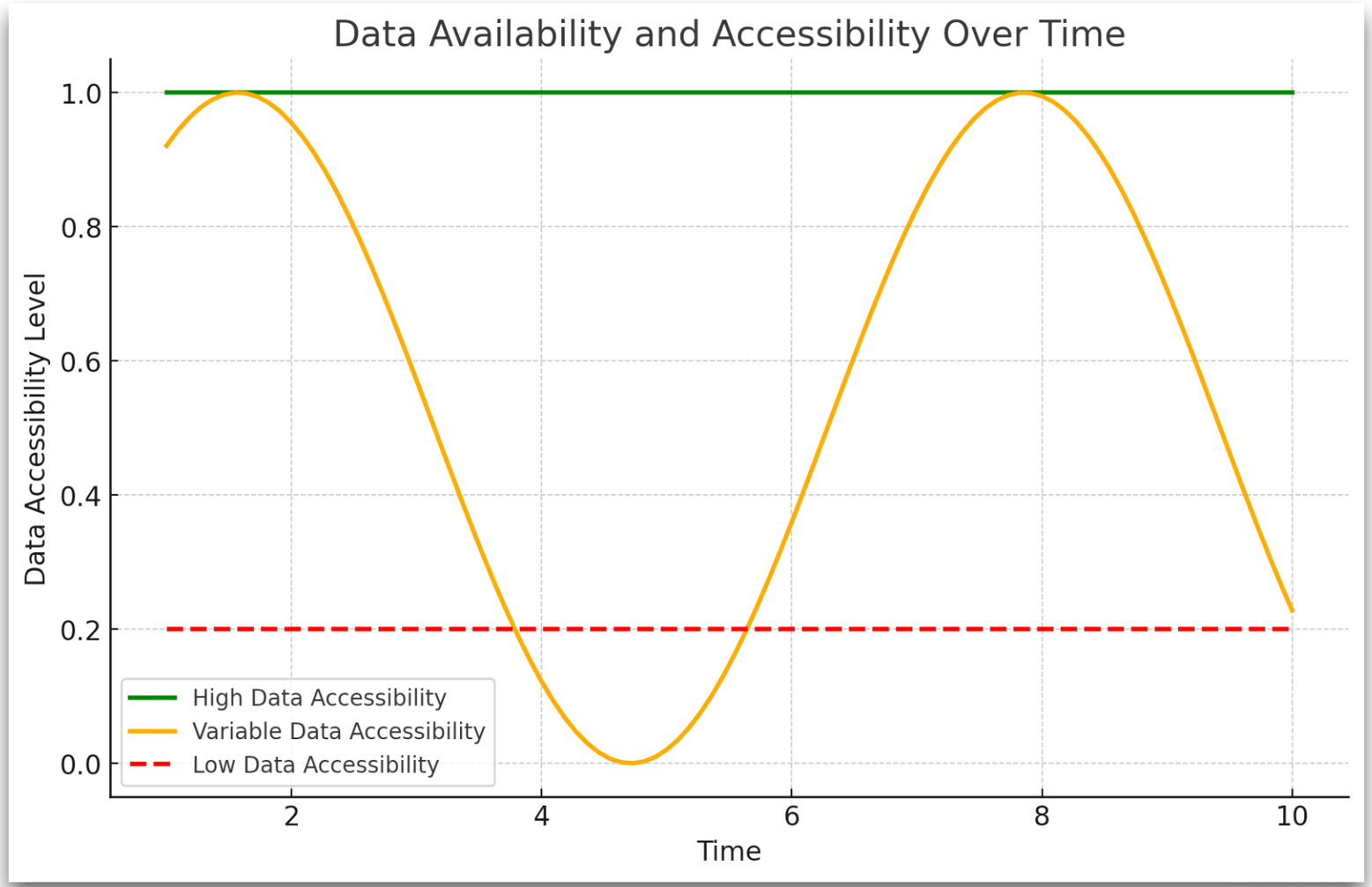
Text Data

Text Data



Data Availability and Accessibility

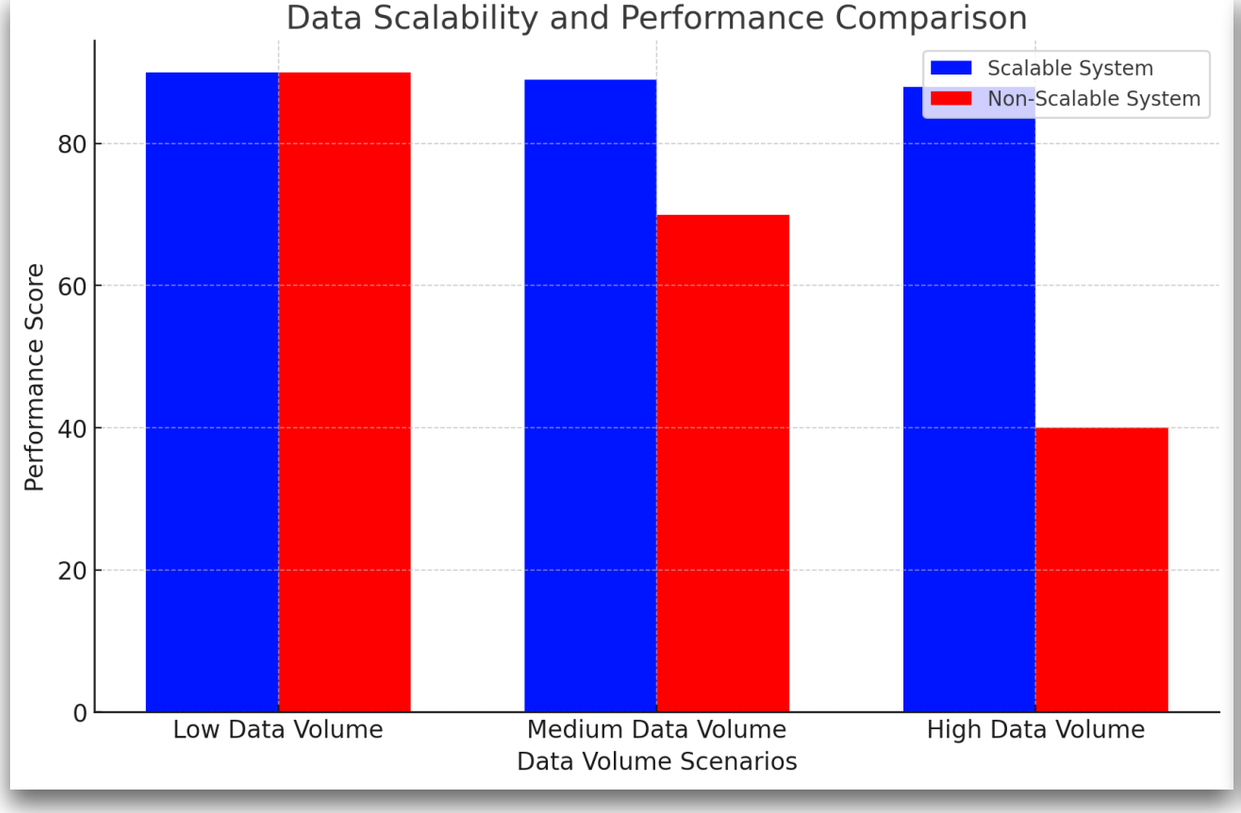
- **Data Silos**: Data stored in isolated systems or departments can be difficult to access and utilize for comprehensive analysis.
- **Lack of Real-time Data**
- **Limited Access to External Data**

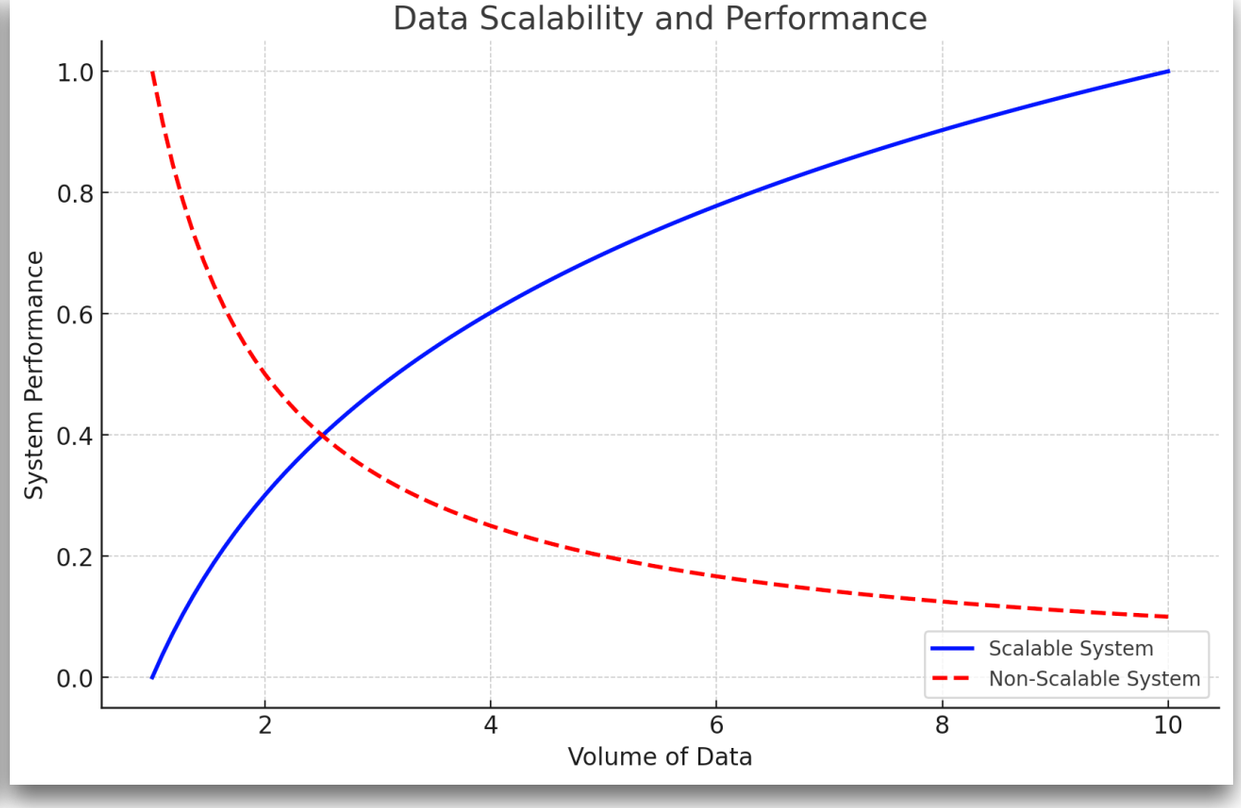


Data Scalability and Performance

- **Handling Large Datasets**: Challenges in processing and analyzing large volumes of data efficiently.

- **Scalability**: Ensuring that data systems and architectures can scale to accommodate growing data needs.

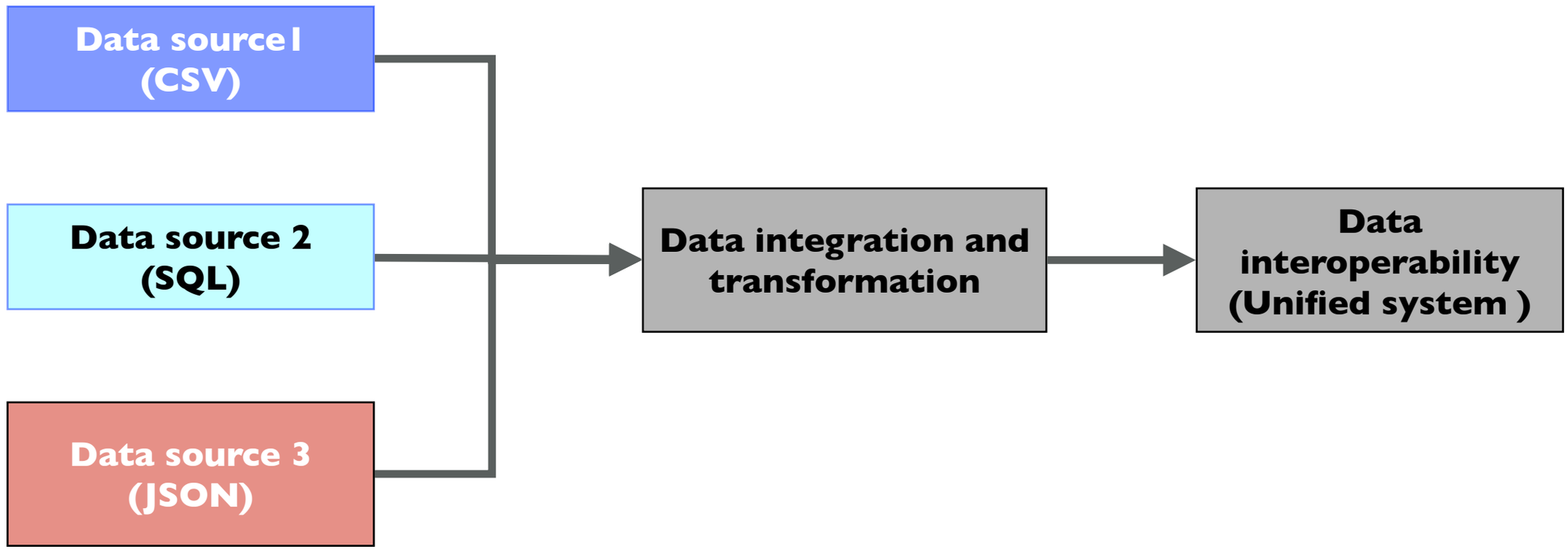




Data Integration and Interoperability

- **Integrating Data from Various Sources**: Challenges in combining data from different sources and formats.

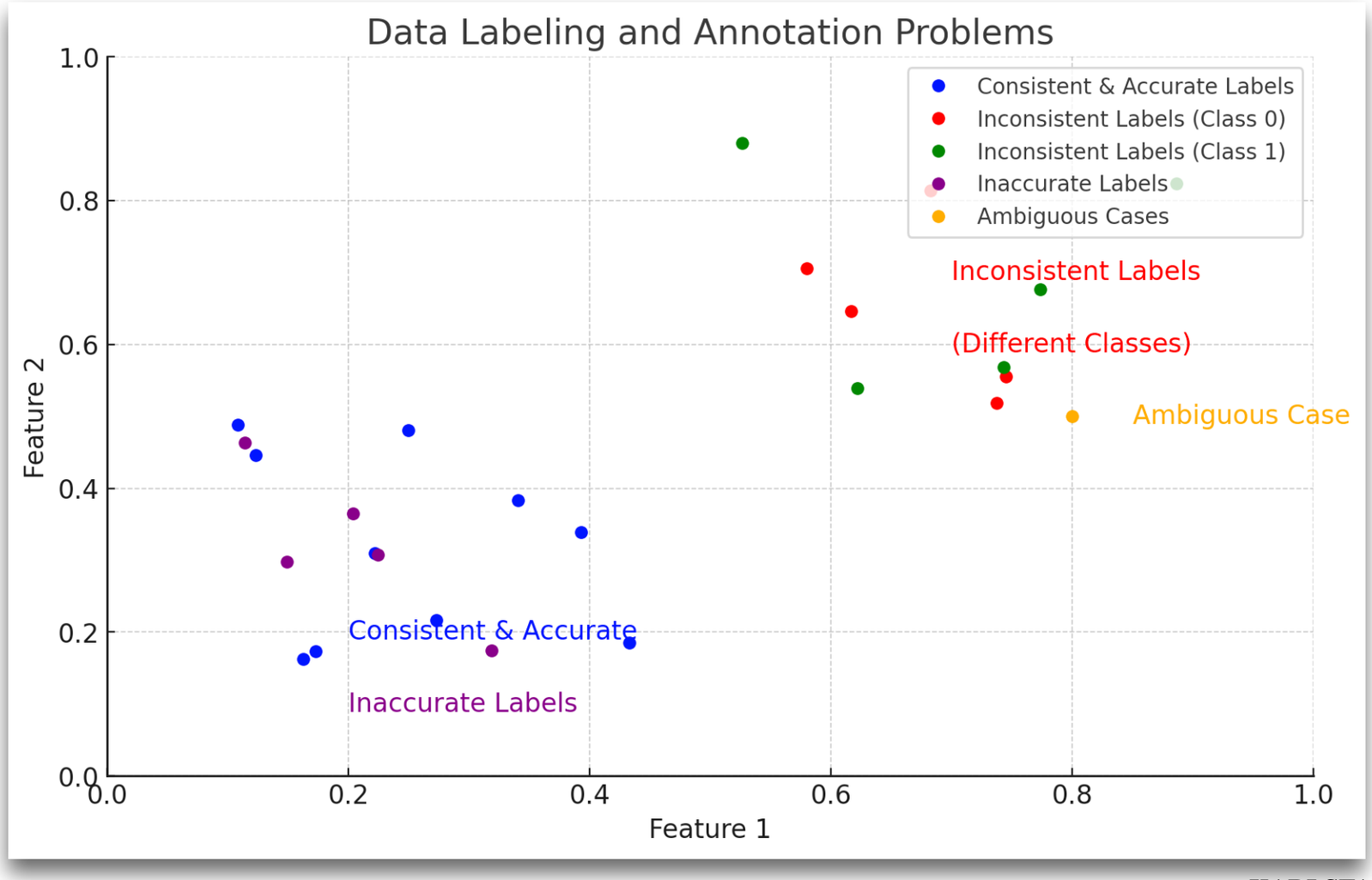
- **Ensuring Data Interoperability**: Making sure that different data systems and software can work together seamlessly.



Data Labelling and Annotation

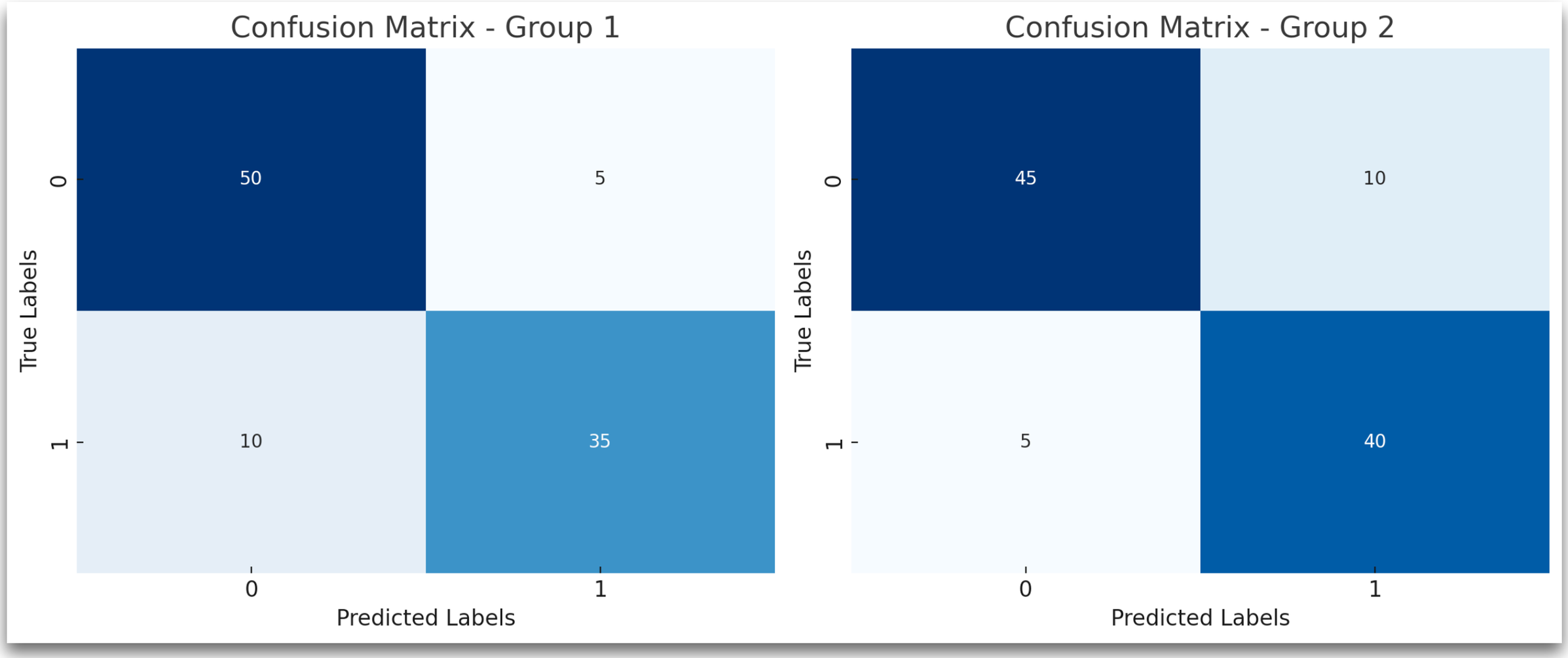
- **Inaccurate Labels**: Incorrectly labeled data can mislead machine learning models.

- **Lack of Expertise**: Insufficient expertise in the domain can lead to poor data labeling and annotation.

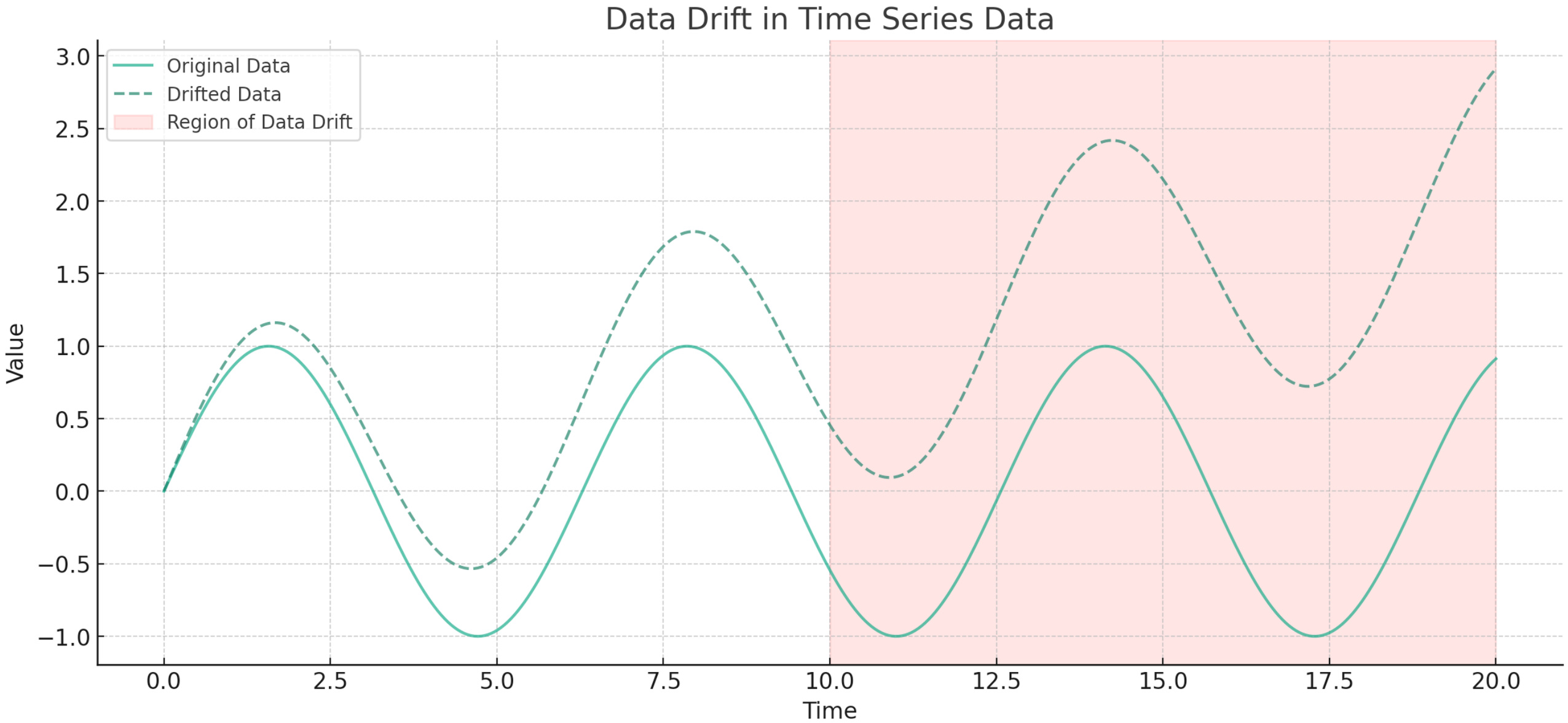


Bias and Fairness

- **Data Bias**: Biased data can lead to biased machine learning models, impacting fairness and objectivity.



Data drift issues

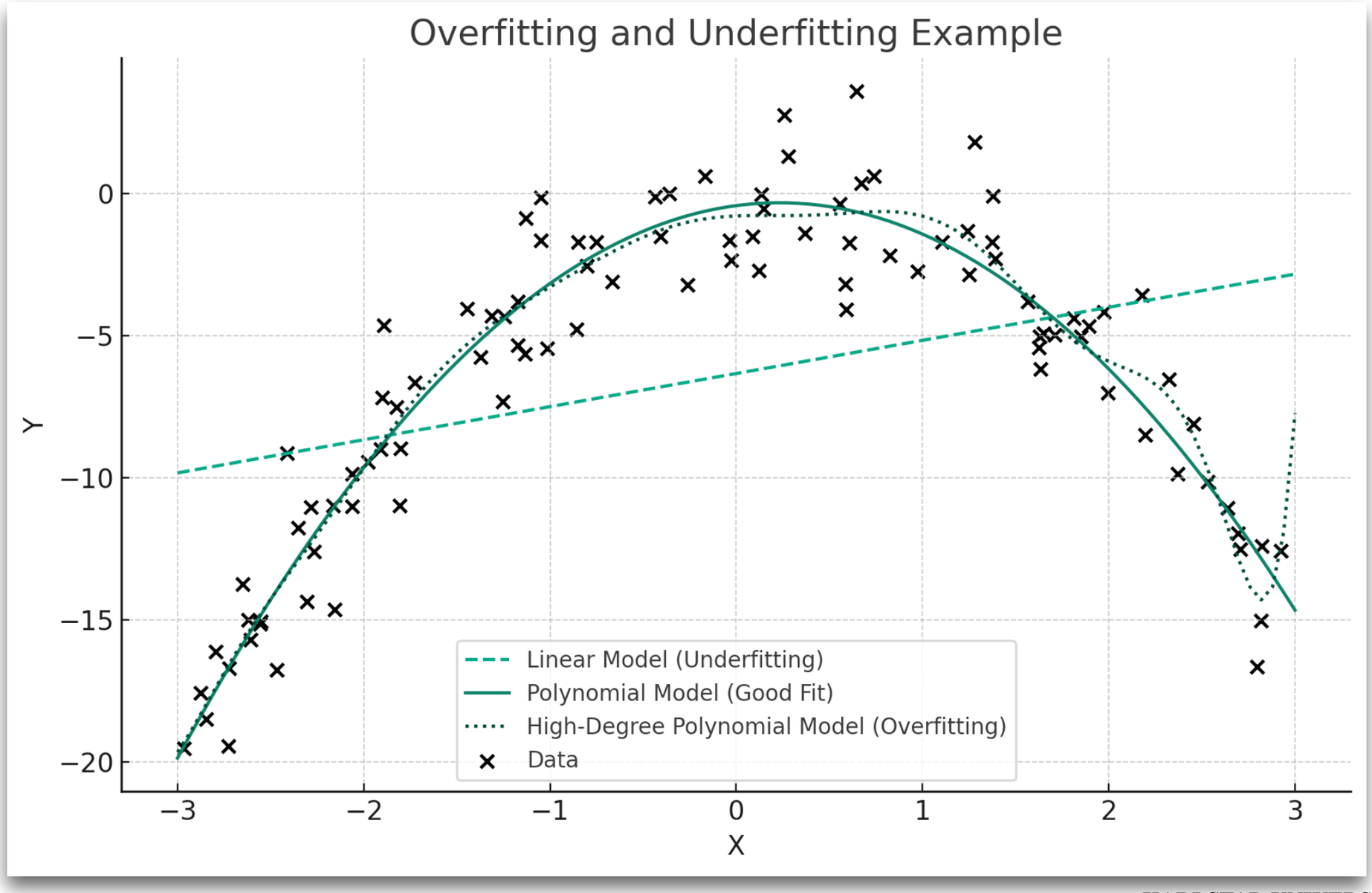


General Model-centric problems

Overfitting and Underfitting

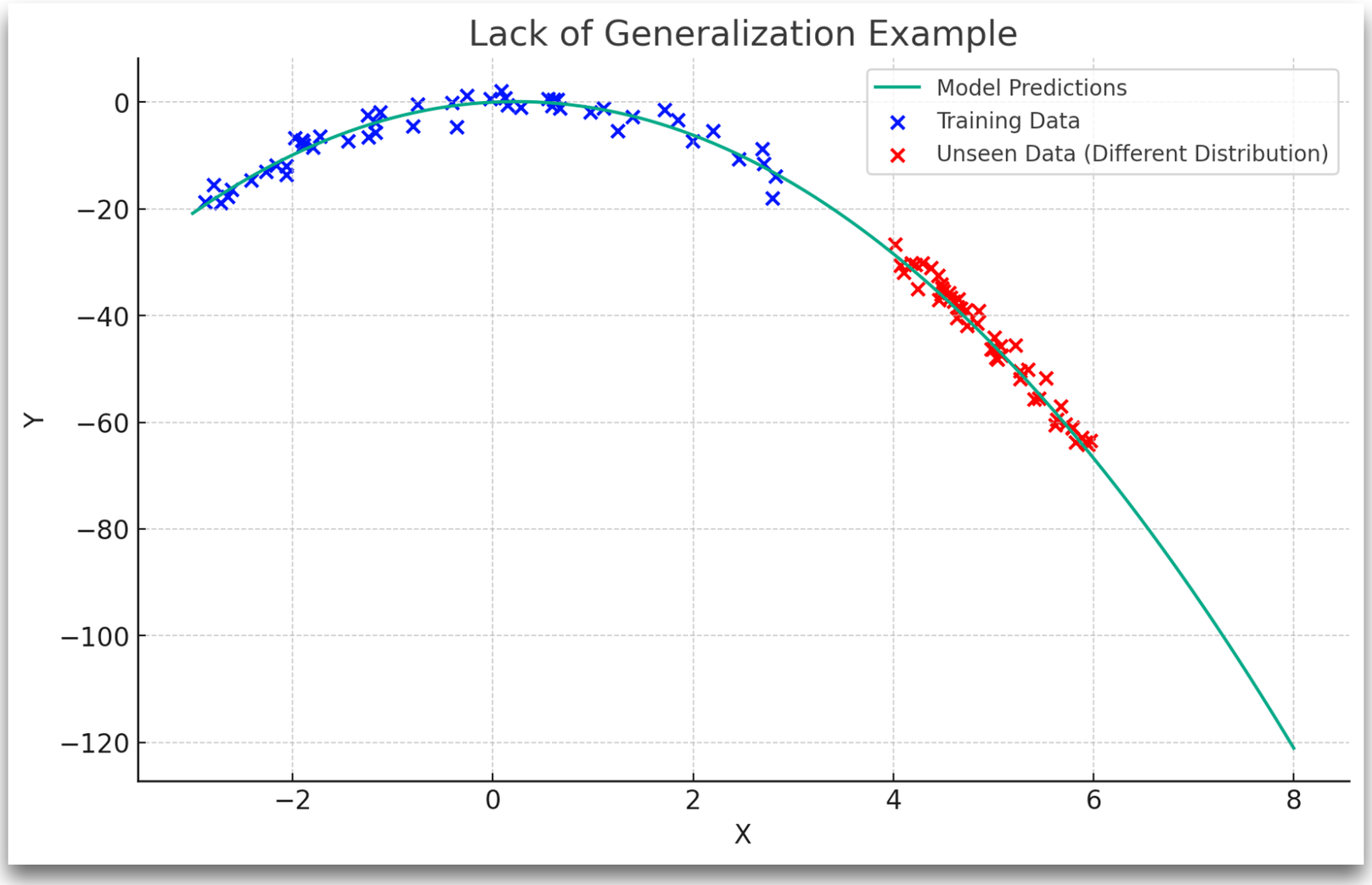
- **Overfitting**: The model performs well on the training data but fails to generalize to unseen data. This often happens when the model is too complex.

- **Underfitting**: The model is too simple to capture the underlying patterns in the data, resulting in poor performance on both training and testing data.



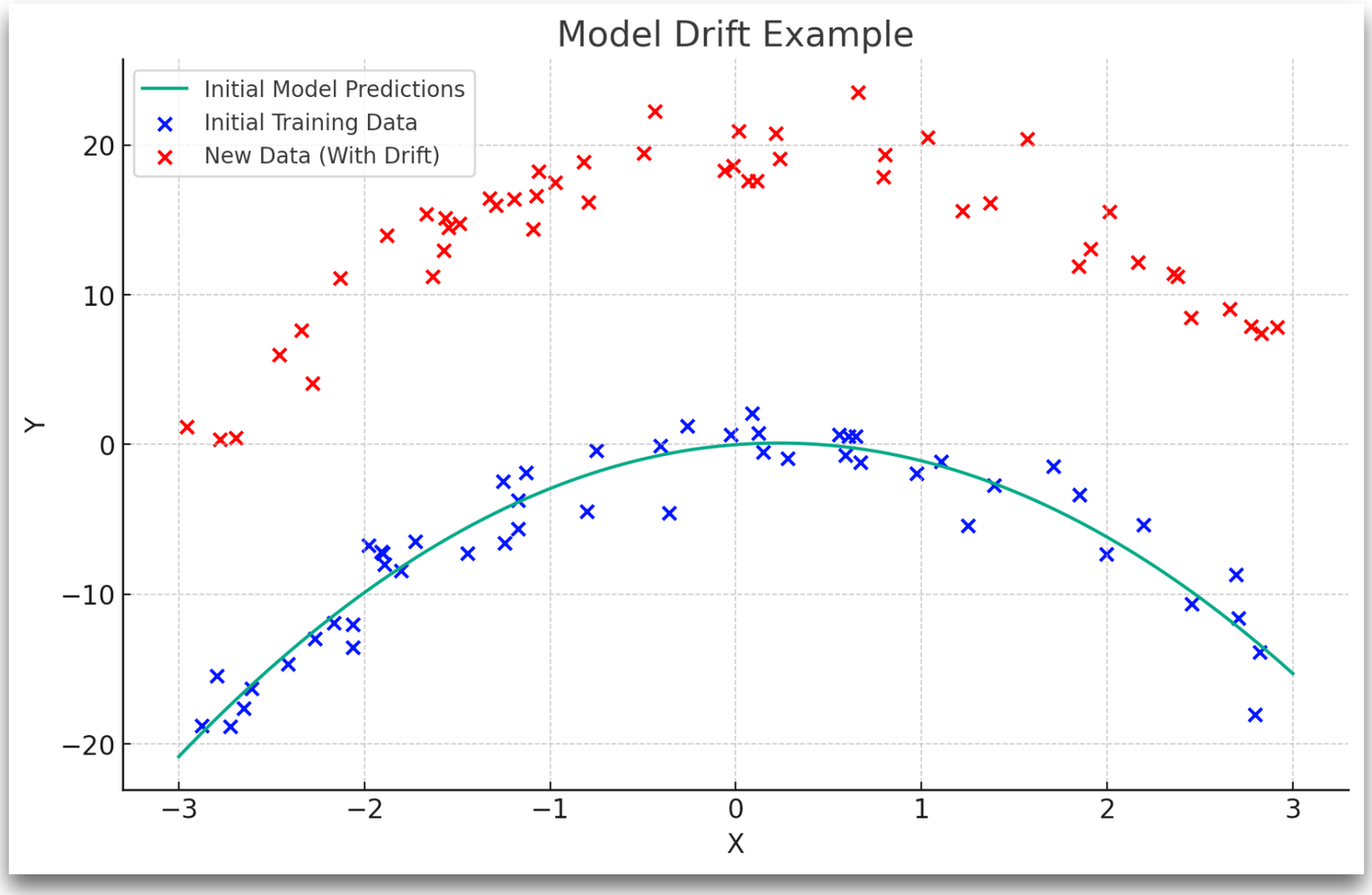
Lack of Generalization

- The model is not able to perform well across diverse scenarios and datasets, limiting its applicability.



Model drift (concept drift)

- **Model Drift:** The model's performance may degrade over time as the underlying data distribution changes, a phenomenon known as model drift or concept drift.



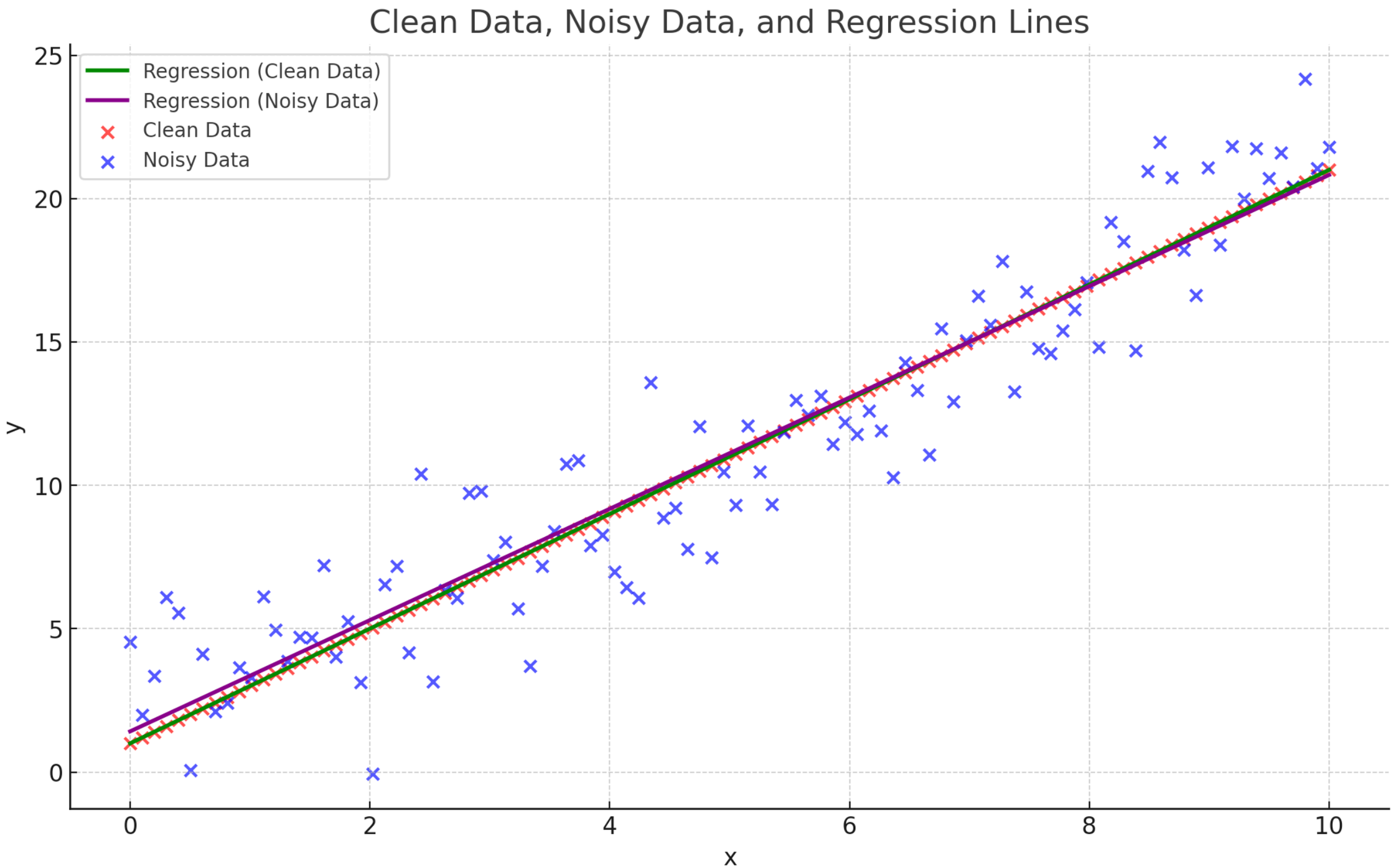
Model Robustness

- **Model robustness** refers to the ability of an AI model to produce accurate and reliable results even in the presence of unexpected or adversarial inputs or conditions.

- Adversarial examples are inputs that are intentionally designed to mislead or confuse an AI model, leading it to produce incorrect outputs.

- **Techniques for model robustness:**
  - **Adversarial training:** a technique that involves training an AI model on adversarial examples to help it learn to better recognize and classify them.
  - **Defensive distillation:** a technique that involves training an AI model to be resistant to adversarial attacks by adding noise to the model's output during training.
  - **Input preprocessing:** a technique that involves preprocessing inputs to an AI model to remove or reduce the impact of potential adversarial inputs, such as by blurring or smoothing images.

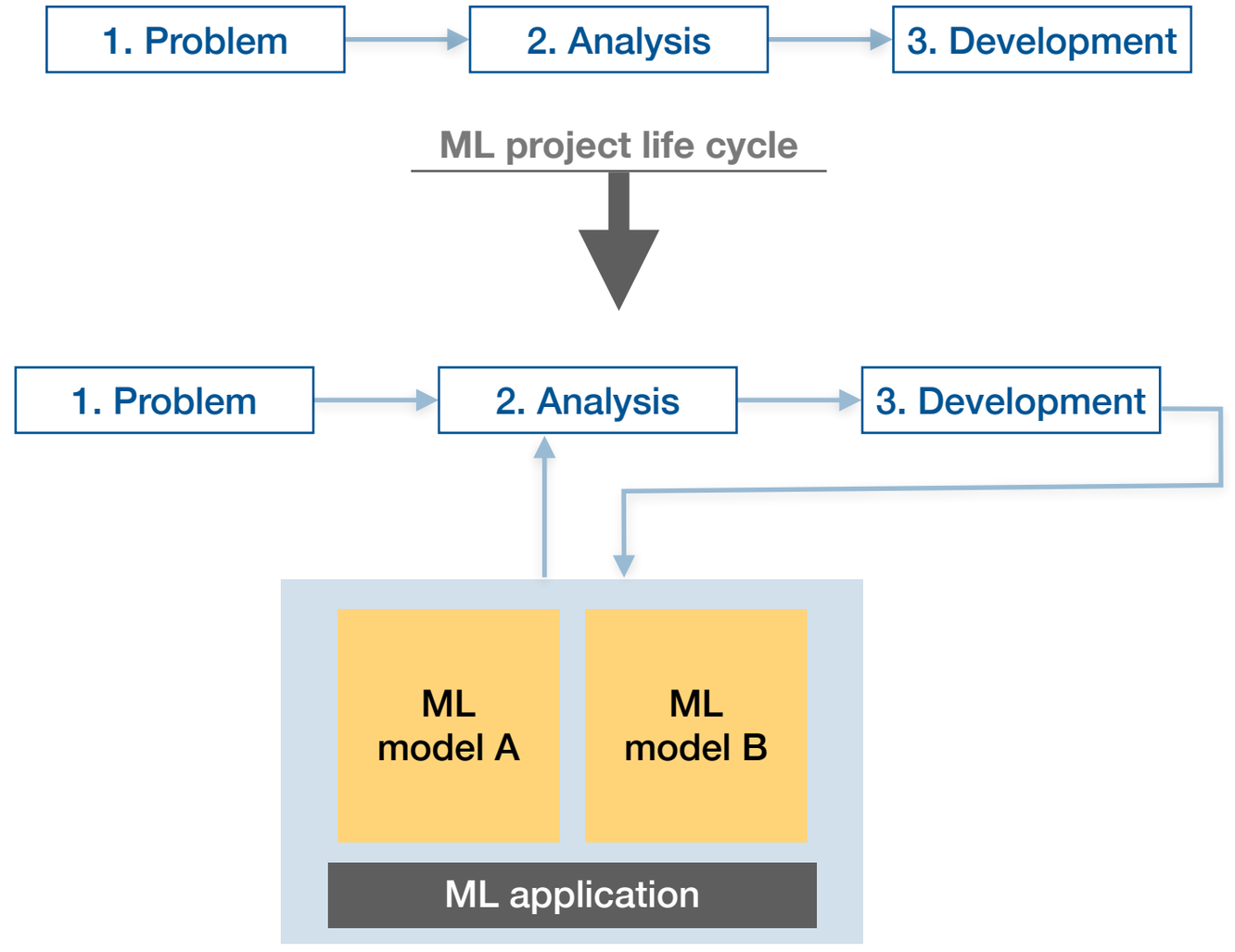
Robustness example



ML from System Perspective

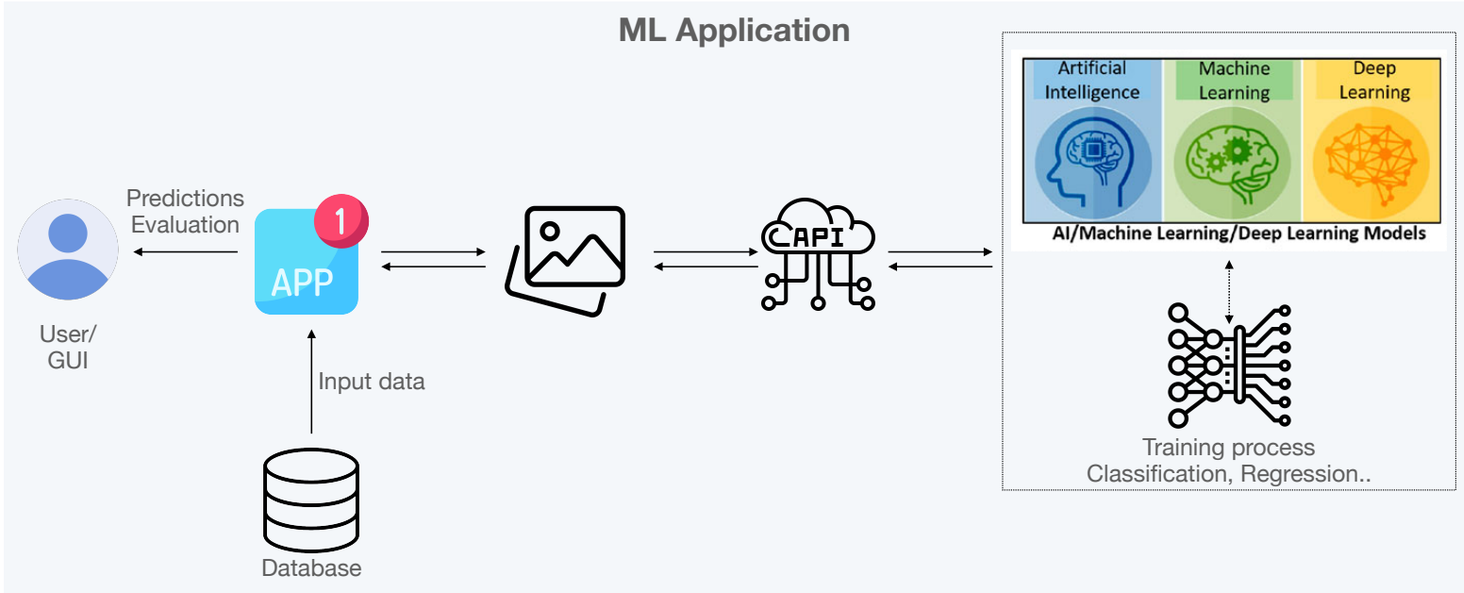
On the ML life cycles

- **Three different lifecycles**
  - ML project life cycle
  - ML application life cycle
  - ML model life cycle
- An ML Model Represents the Fundamental Estimator
  - Eg. When it serves as a predictive tool generating daily sales forecasts.
- The ML Application, on the other hand, encompasses of supplementary components and functionalities, alongside its core ML models.

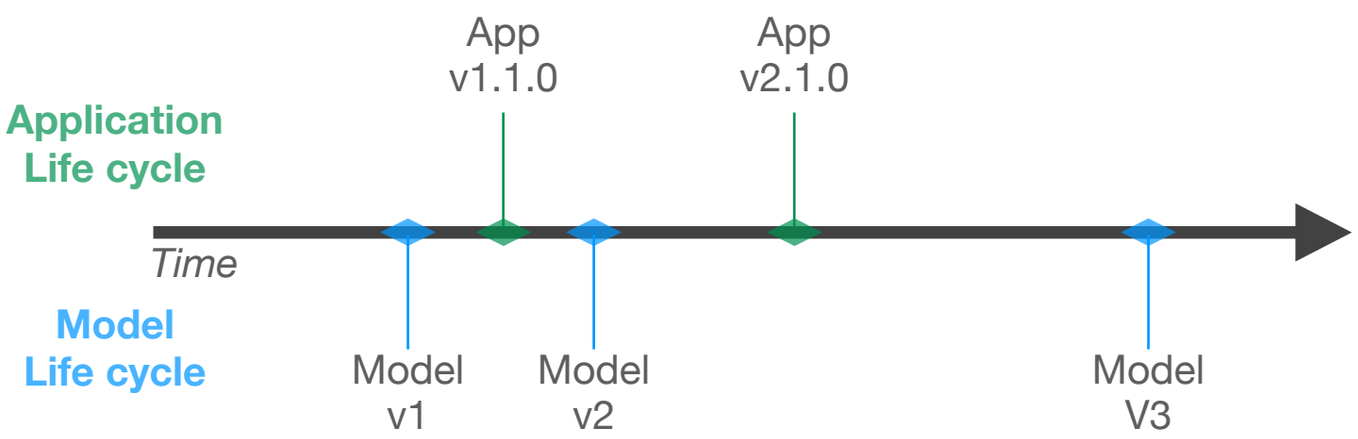


ML App

- **Database**, for storing extra features, and logging model outputs
- **GUI:** for admin users, to configure and troubleshoot the app.
- **API**, through which the app can communicate with the outside world in a consistent and secure manner,



- In best practice the ML model is not locked to the ML application so that each can go its own way, which is the philosophy of the so-called "microservice architecture”



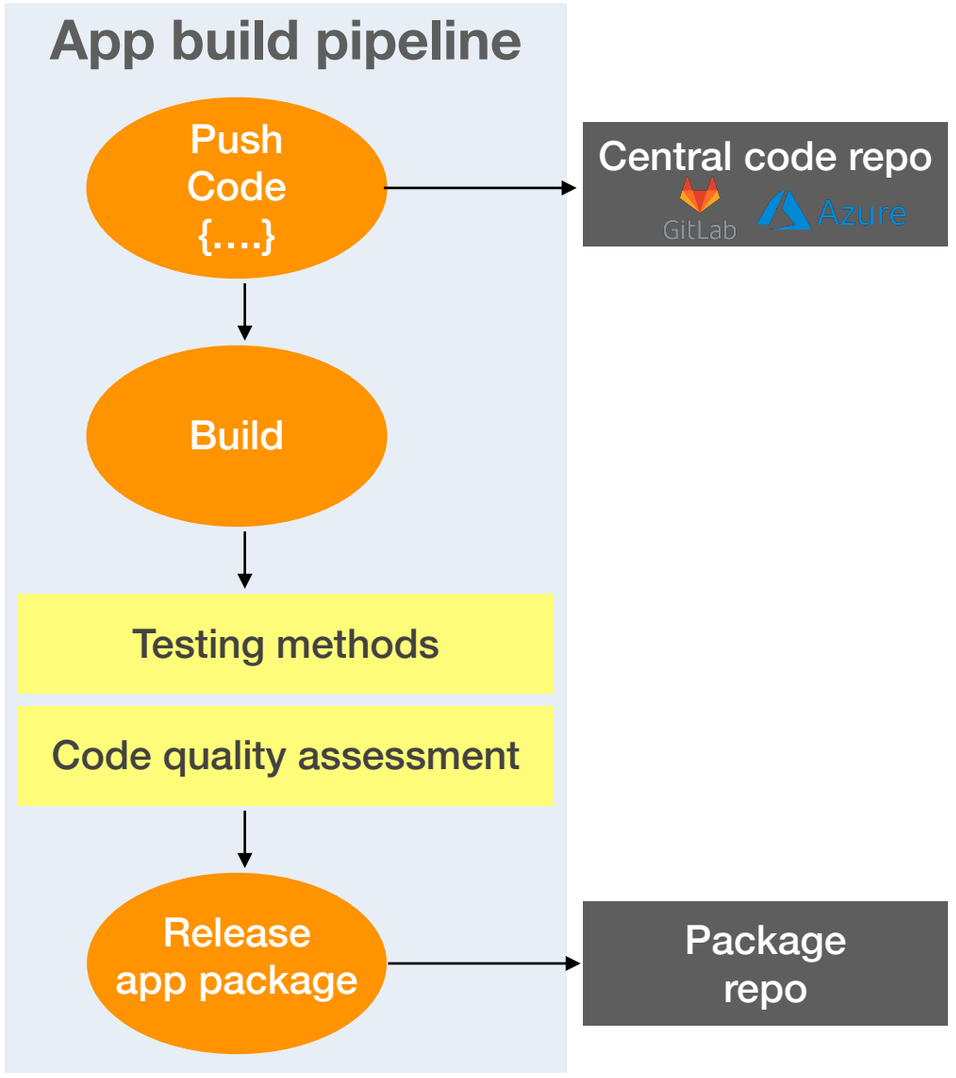
App Building Process

- The app build process follows the classical DevOps software build approach.

- It initiates by retrieving our scripts from a centralized code repository, which serves as our code management hub.

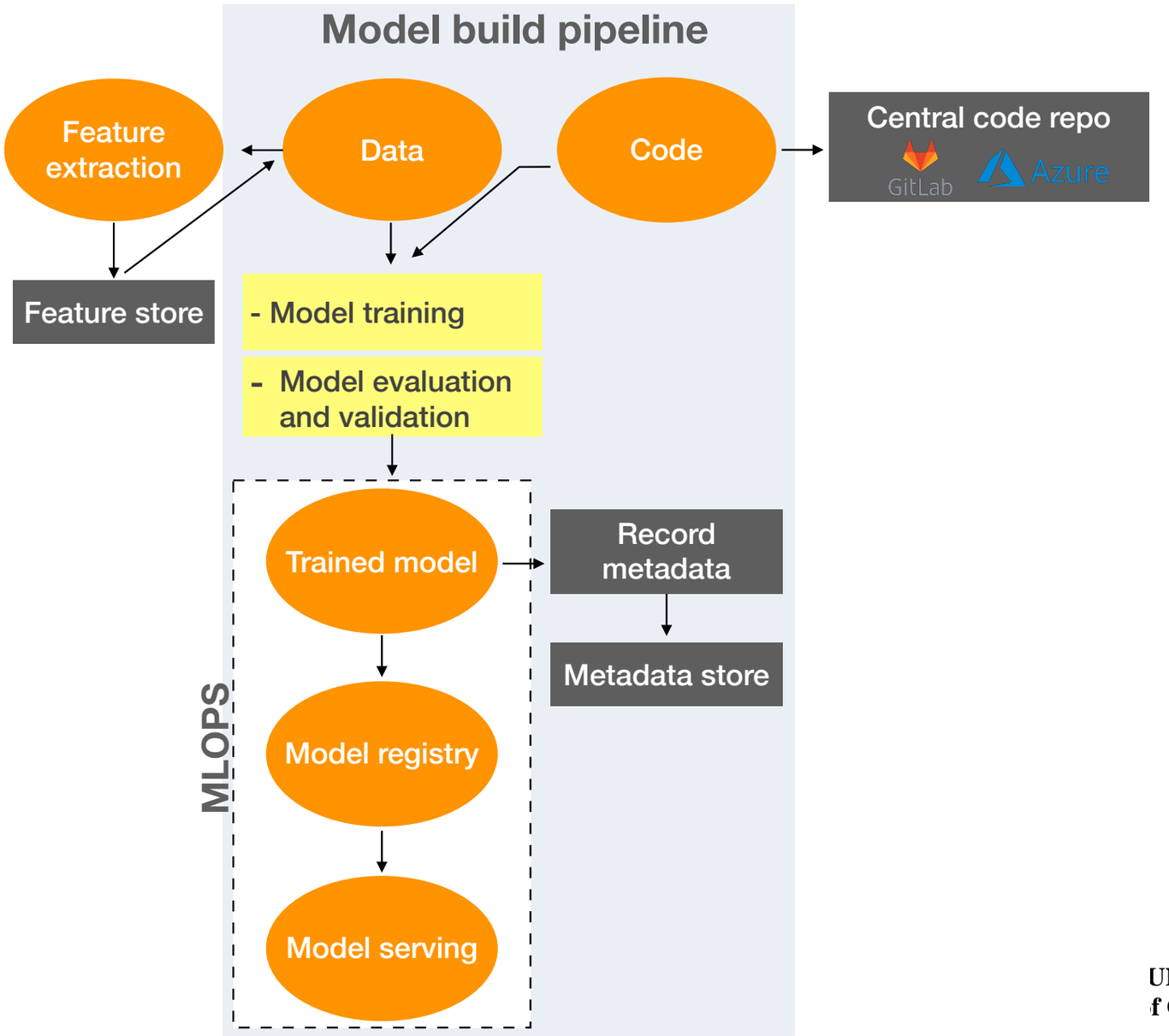
- Automated transformations are then applied to prepare the app for deployment.

- The resulting package is stored securely in a dedicated repository.



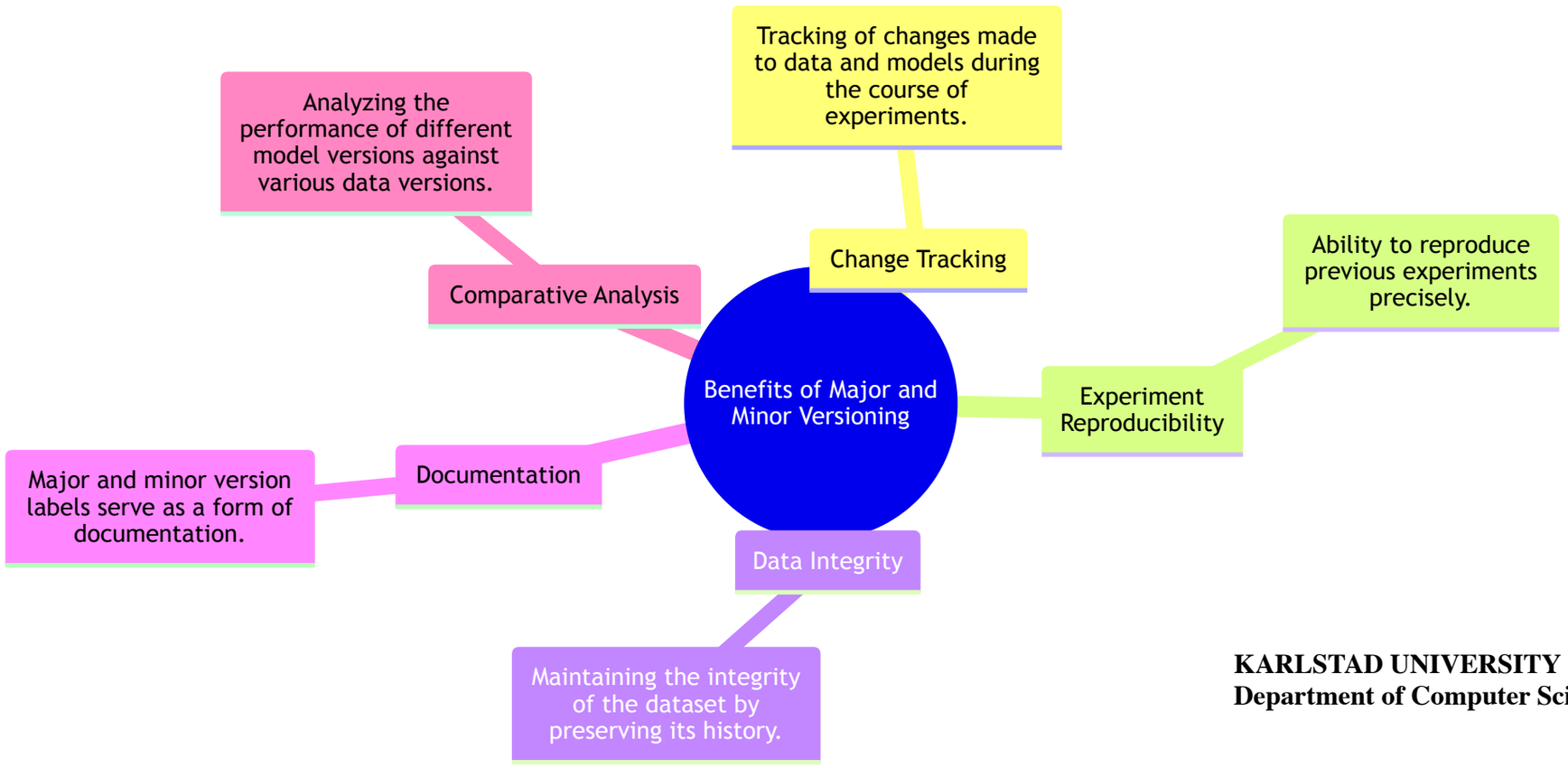
ML Build Process

- The model build pipeline, also known as the model training pipeline is different.



Versioning Training Data

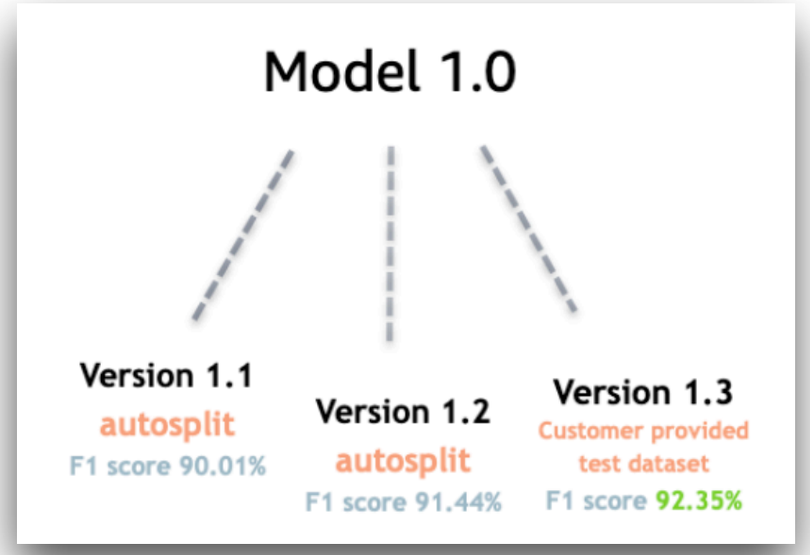
- **Major versioning:**
  - Indicates significant changes in either the data or the model.
  - Substantial alterations to the dataset, such as adding or removing a significant portion of data, or fundamental changes to the model architecture or training process.
- **Minor versioning:**
  - Signifies smaller, incremental changes to the data or model.
  - Often involve smaller modifications like adding new features, optimizing data preprocessing steps, or fine-tuning model hyperparameters.



Versioning ML Models

- Like data, ML models also versioned to ensure reproducbility and enabling rollbacks.

- Usually matches training data version but not always



- **Example:**
  1. Model V 1.0 is our initial model (Random Forest)
  2. Model V 1.1 is the same model fine-tuned on Data V 1.1
  3. Model V 2.0 is now a XGBoost Model fine-tuned Data V 1.2
  4. Model V 2.1 is the XGBoost model fine-tuned on Data V 2.0

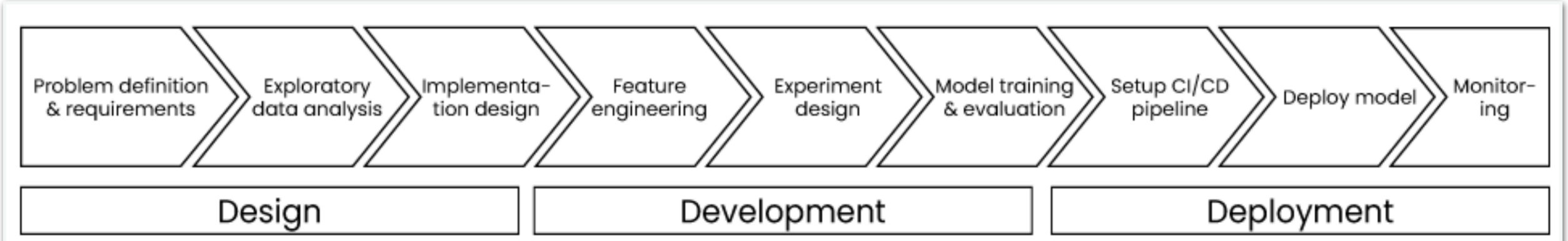
MLOps: Bridging the Gap between ML and Operations

Machine Learning Operations

- **MLOps:** is the set of practices to **design**, **deploy** and **maintain** machine learning in production **continuously**, **reliably**, and **efficiently**.

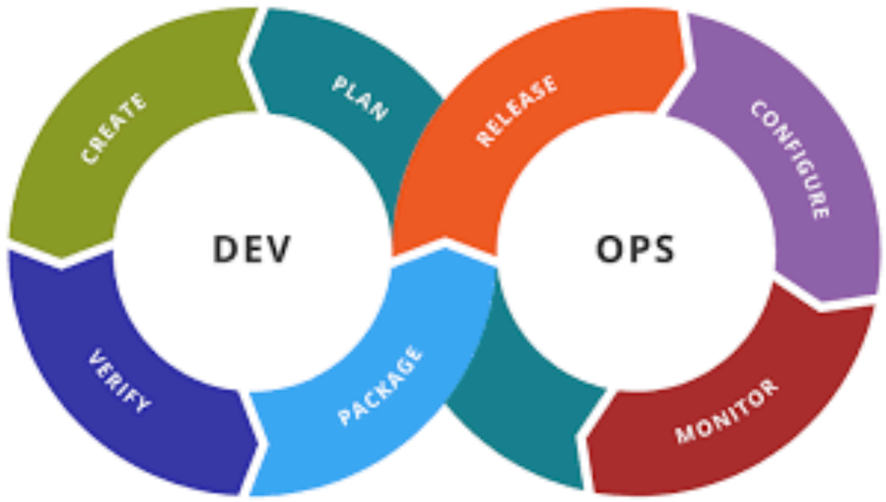
- **Focus on 'In Production':** MLOps centers around the deployment and ongoing management of ML models in real-world, production environments.

- **Full ML Lifecycle:** It encompasses the entire ML lifecycle, including design, development, deployment, and maintenance in production.

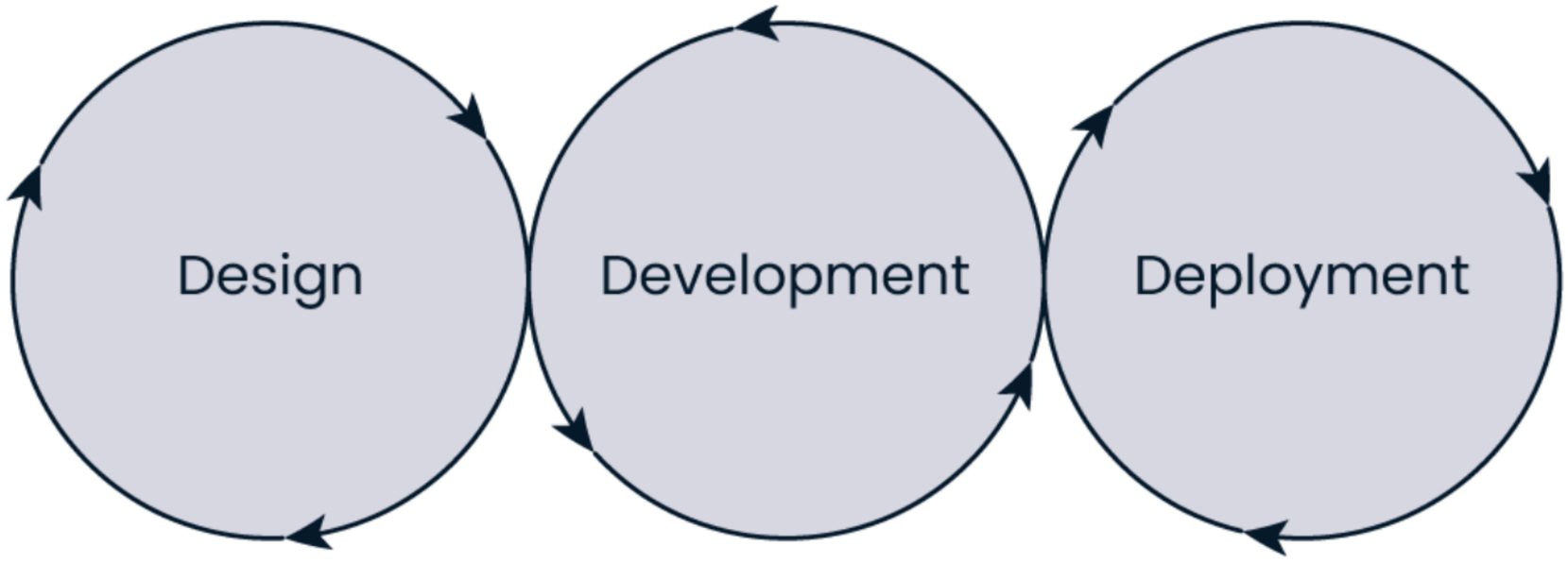


- **Origins in DevOps:** MLOps shares its roots with DevOps, which stands for Development Operations a set of practices and tools used in software development to ensure continuous, reliable, and efficient software development.

- **DevOps Principles:** DevOps principles, such as accelerating development processes and fostering collaboration between development and operations teams, are applied to MLOps to streamline the deployment and maintenance of ML models.



MLOps Lifecycle



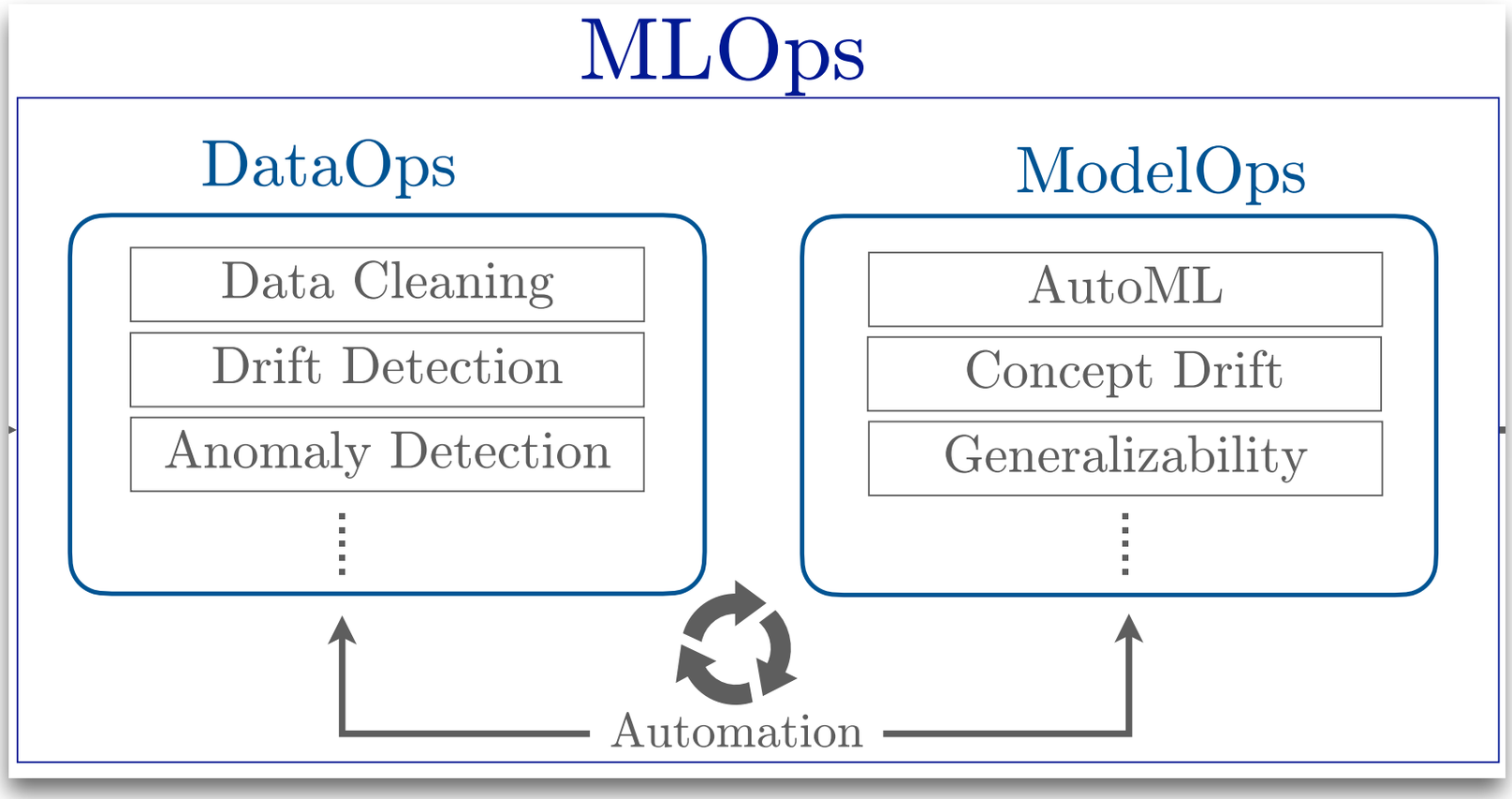
- **ML Lifecycle in MLOps:**
  - **Design:** Project conceptualization and goal-setting.
  - **Development:** Model building, training, and evaluation.
  - **Deployment:** Model goes live in production.
- Phases often overlap.
- Continuous evaluation with stakeholders is key.

Building for Scale: Automation First

- To build scalable ML systems, our focus is on streamlining processes, embracing best practices, and automation:
  - **Automation-First Approach:** Automation is our foundation to prevent hidden technical debt and unlock scalability. It allows us to maintain and update ML systems efficiently.
  - **MLOps for High-Value Solutions:** MLOps is our key to delivering valuable ML solutions that receive frequent and reliable updates. It ensures our models perform optimally in production.
  - **Transparent and Reproducible Processes:** We prioritize transparency and reproducibility through established methodologies like **CRISP-DM** and **Microsoft's TDSP**. These frameworks guarantee quality and reliability in our ML systems.

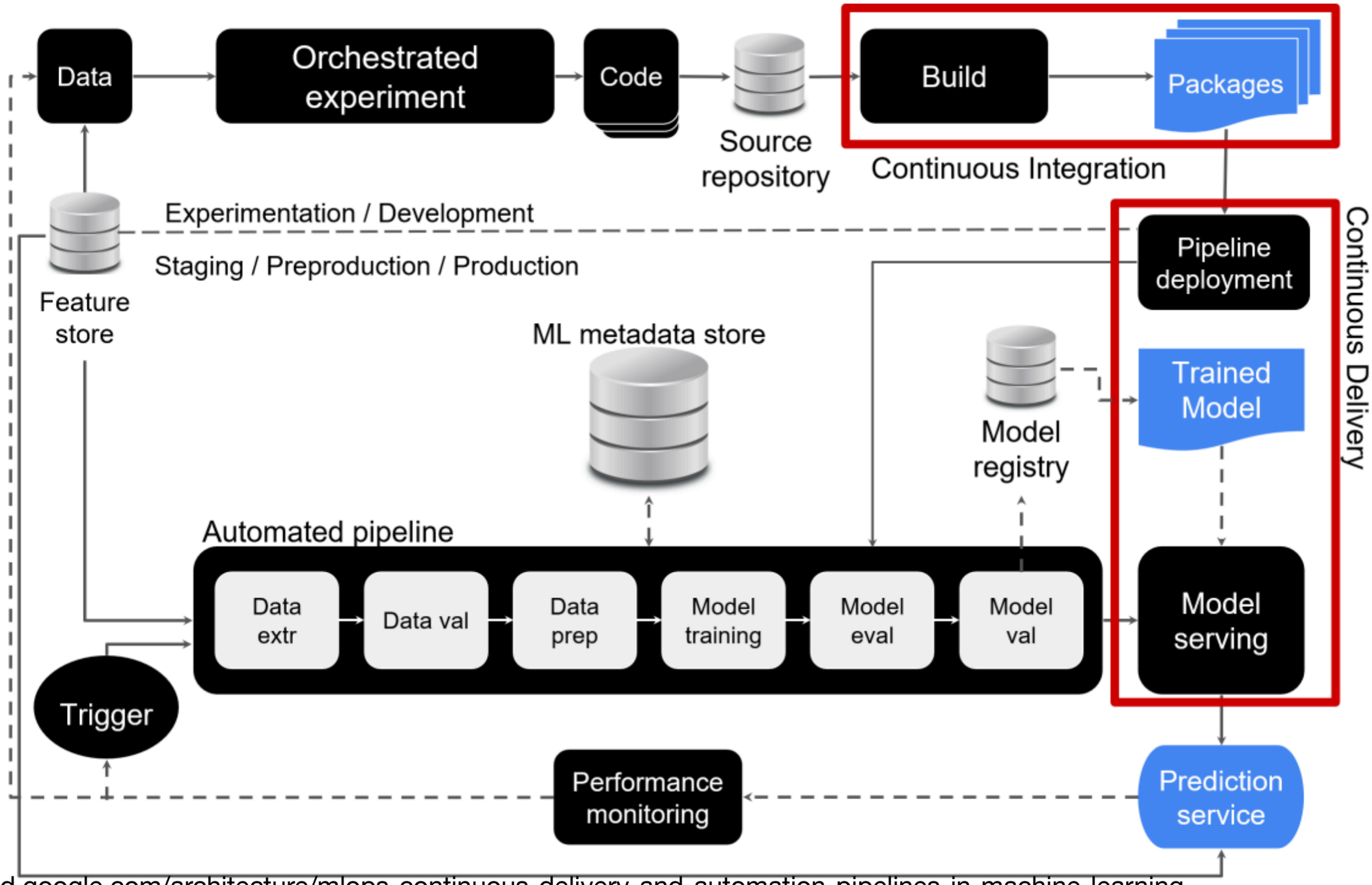
Towards Robust ML Systems in Production

- Robust performance is an essential requirement to build **trustworthy AI systems** (according to EU guidelines)
- **DataOps**: end-to-end data processing operations in production
- **ModelOps**: the set of operations that are performed on the learning task of the ML model
- **Automation**: the engine that drives and coordinates the overall operations



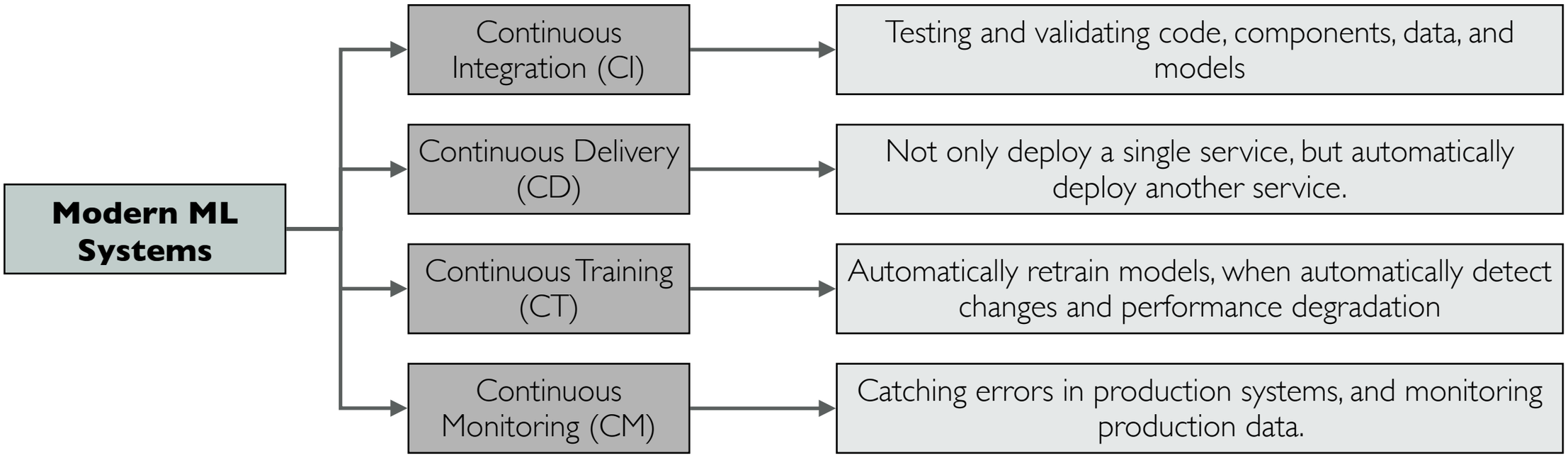
Fully Automated MLOps Architecture

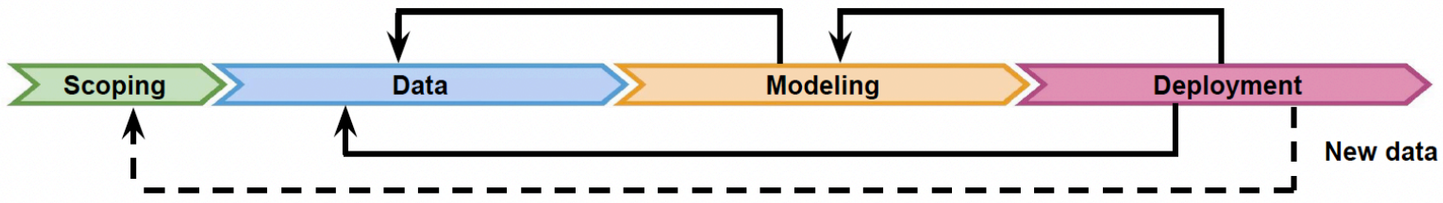
- A fully automated MLOps solution has **<u>CI/CD capabilities</u>** as well as two more capabilities: **<u>continuous monitoring</u>** and **<u>continuous training</u>**.

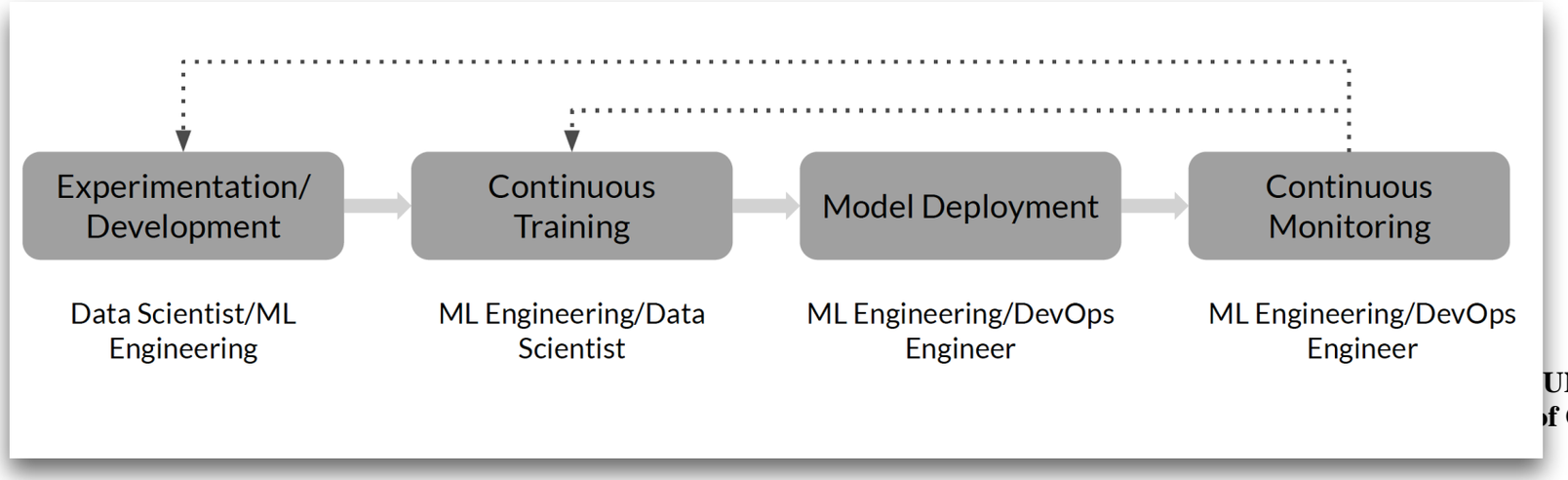


https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

Modern ML Systems in Production

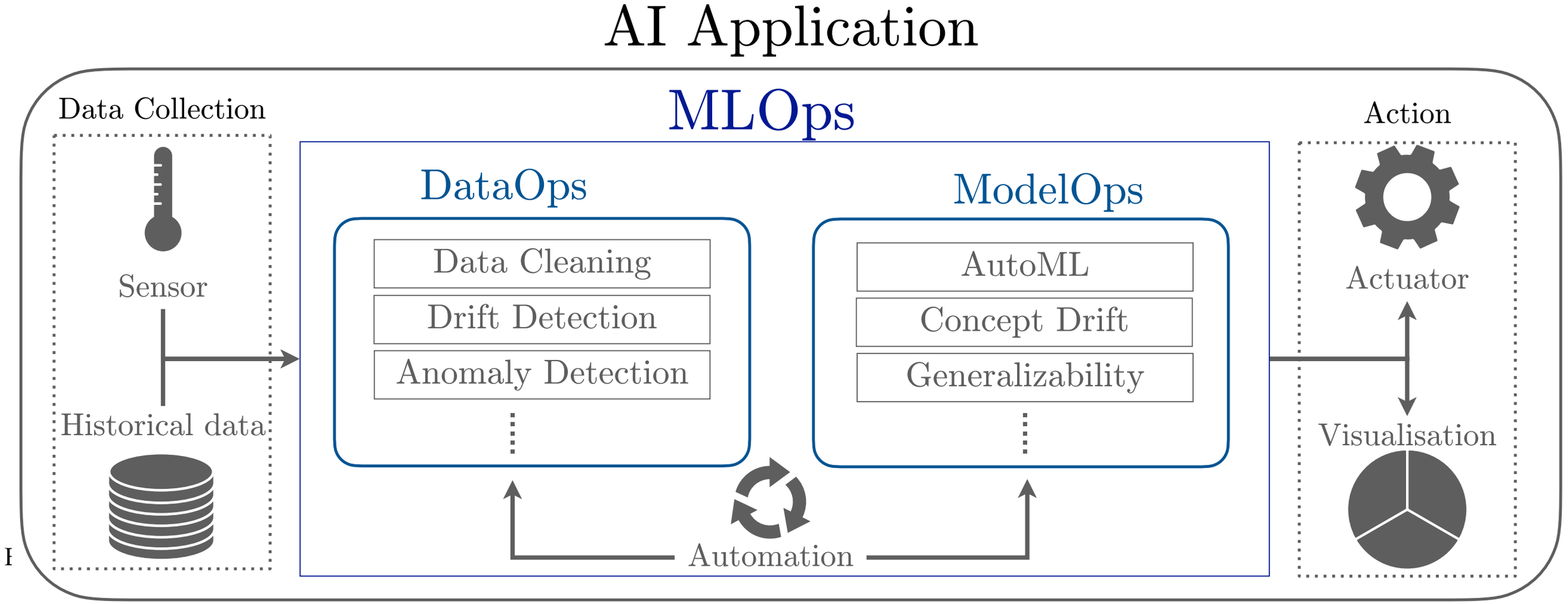




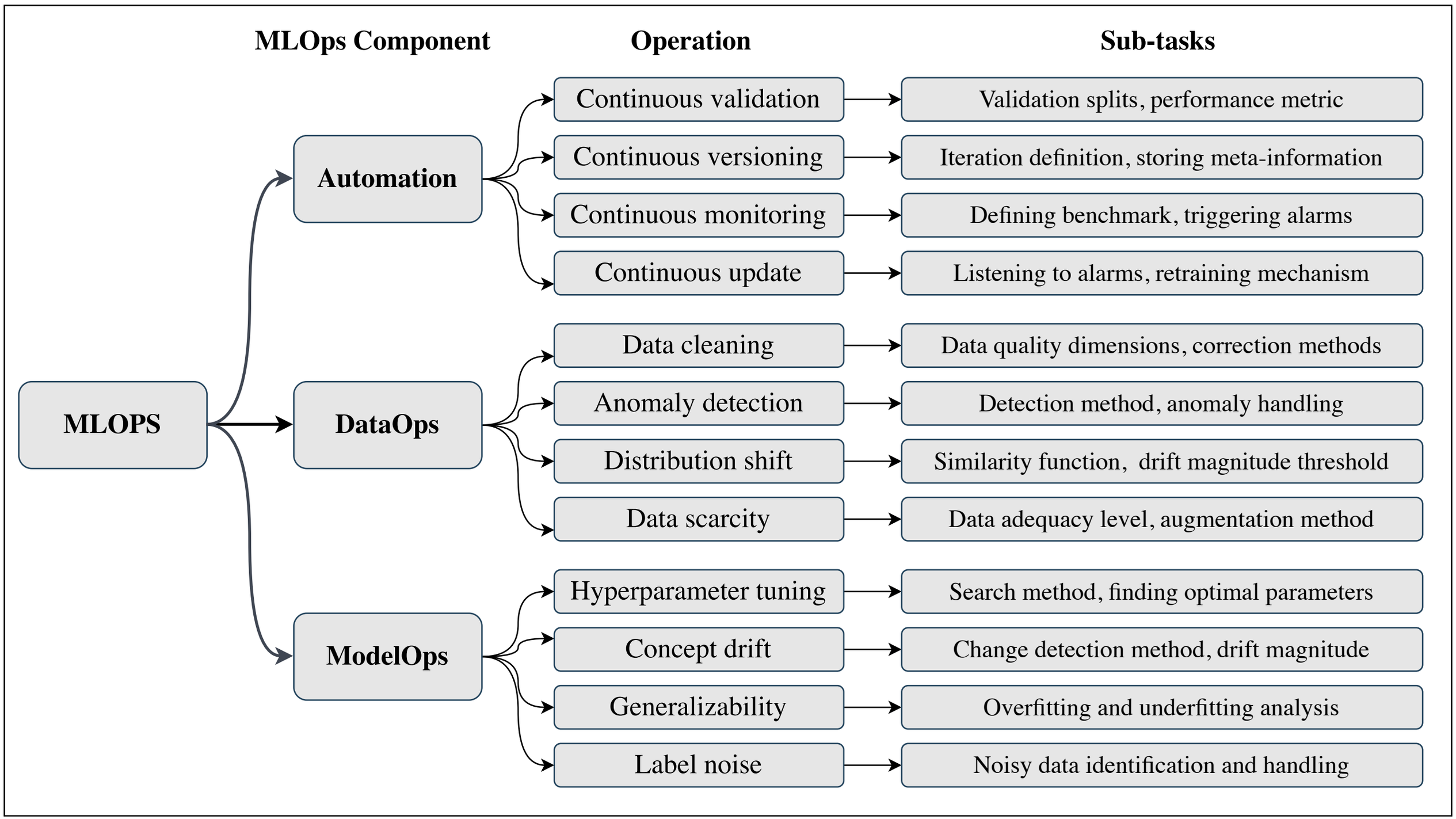


Continuous Robustness

- **Continuous ML system (MLOps)**: DataOps+ML ModelOps+automation
- Necessary shift from accuracy-driven to **robust-driven** systems
- **Robust ML system in production**: Maintaining high performance of ML in real-life deployment. From accuracy-driven to robust-driven Systems



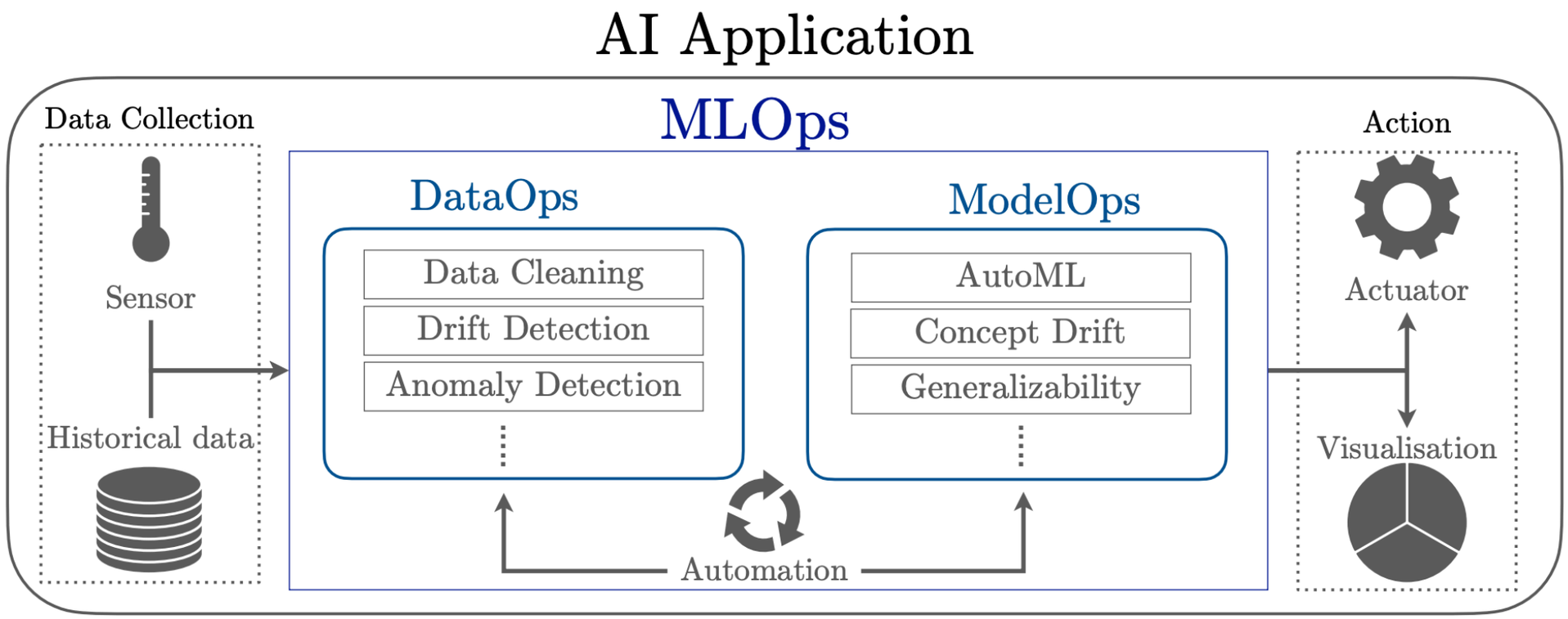
MLOPs Dimensions



Deploying AI at Scale

- **Continuity and automation .. Towards continuous everything**

- **Several Challenges:**
  - **Data Quality and Quantity:** Large-scale deployment requires a huge amount of high-quality, labeled data
  - **Model Performance:** The accuracy of AI models can degrade at scale
  - **Model Explainability and Trust:** The deployment of AI systems at scale raises important ethical and legal concerns
  - **Integration with existing systems:** can be challenging due to compatibility and technical difficulties.
  - **Maintenance and Updating:** AI models need to be maintained and updated regularly to ensure that they continue to perform well.

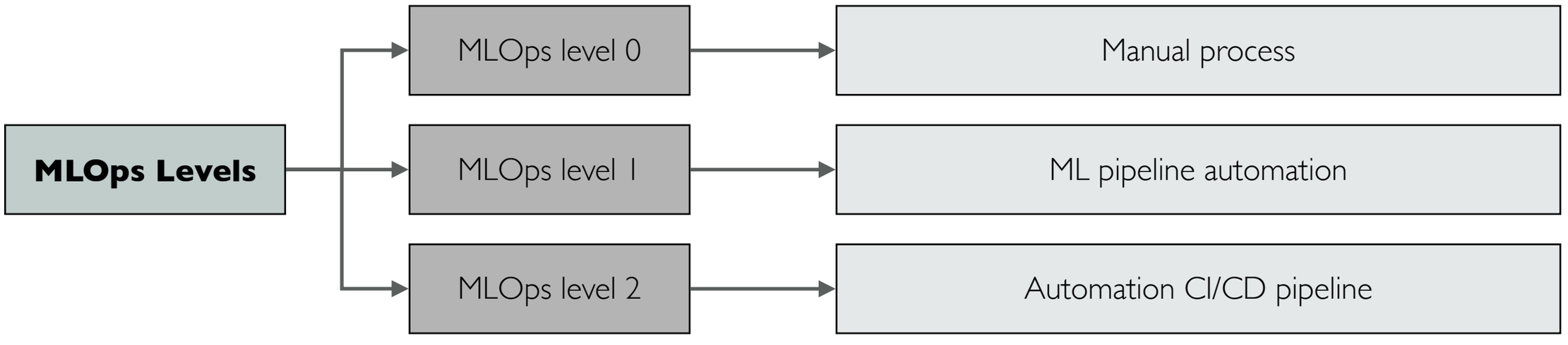


Bestoun S. Ahmed,

MLOps Levels in Production and Service

MLOps Process' Maturity

- The level of automation of ML pipelines determines the maturity of the MLOps process
- As maturity increases, the available velocity for the training and deployment of new models also increases
- Goal is to automate training and deployment of ML models into the core software system, and provide monitoring



Customization With the Current MLOPs Tools

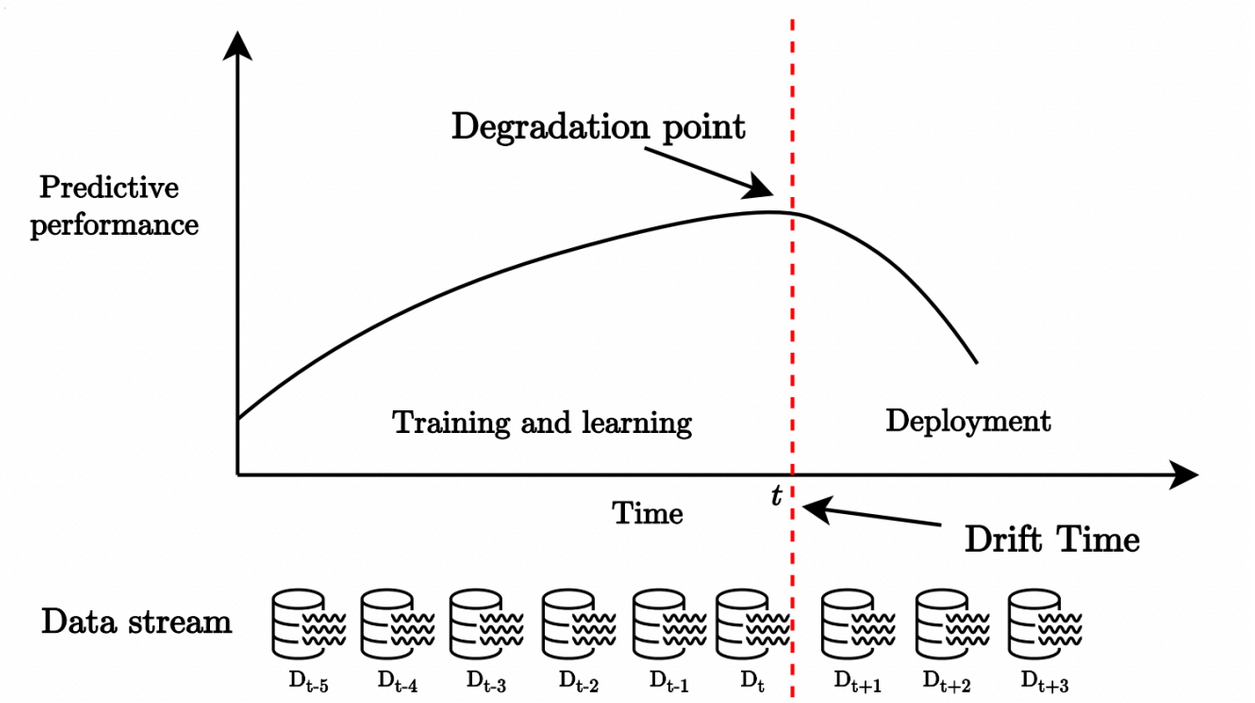
- Summary of the built-in features of MLOps tools

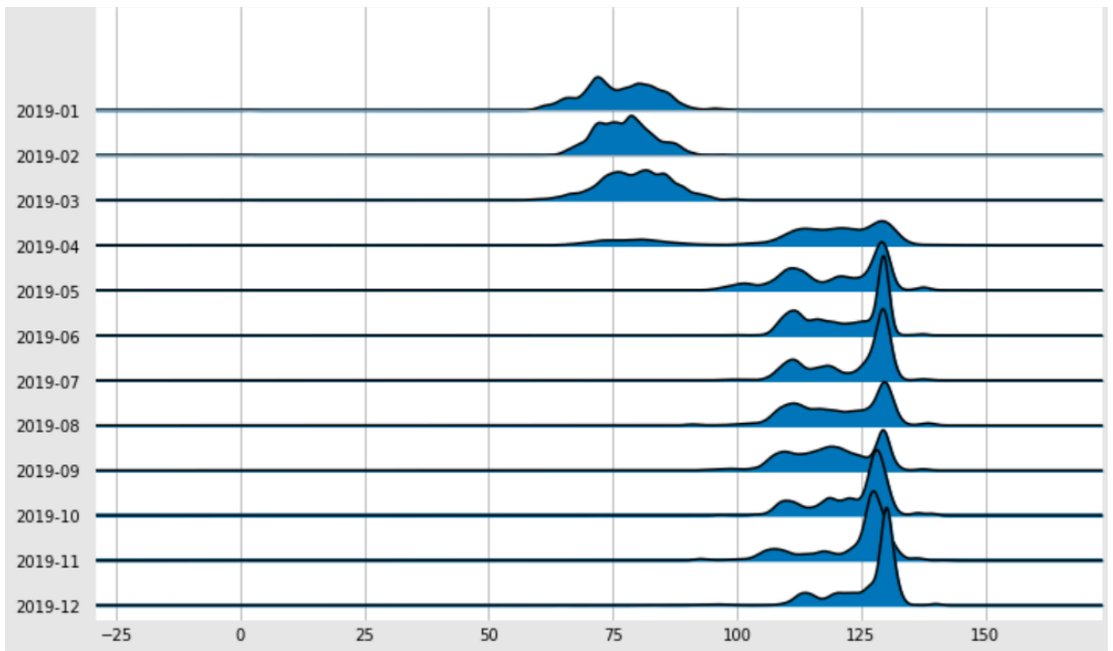
	Operation	MLOps tool				
		AWS	Azure	Vertex AI	MLFlow	TFX
Automation	Continuous validation	✓	✓	✓	✓	✓
	Continuous versioning	✓	✓	✓	✓	✓
	Continuous monitoring	✓	✓	✓	✓	✓
	Continuous update	✓	✓	✓	✓	✓
DataOps	Data cleaning	✓	✓	✓	✓	✓
	Anomaly detection	✓	✓	✓	-	✓
	Distribution shift	✓	✓	✓	-	✓
	Data augmentation	-	-	-	-	-
ModelOps	Hyperparameter tuning	✓	✓	✓	✓	✓
	Concept drift	✓	-	-	-	-
	Generalizability	-	✓	-	-	-
	Label noise	✓	-	✓	-	-

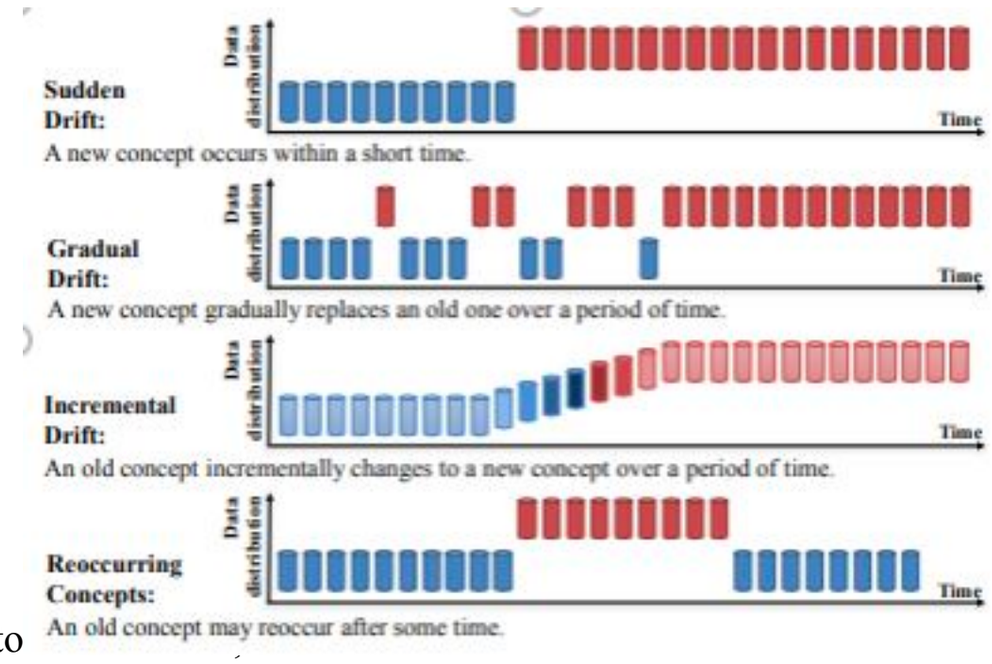
MLOps tools may not offer enough customization options to meet the unique needs and requirements of different organizations and use cases.

Our Current Progress and Future Approaches

Data Drift and Model Degradation







Besto

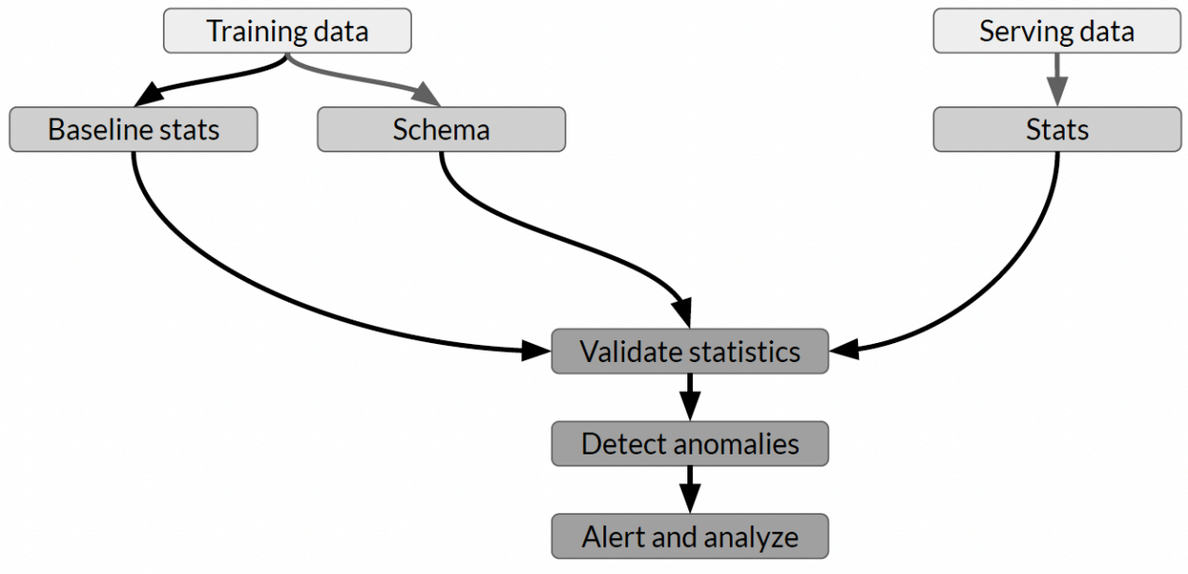
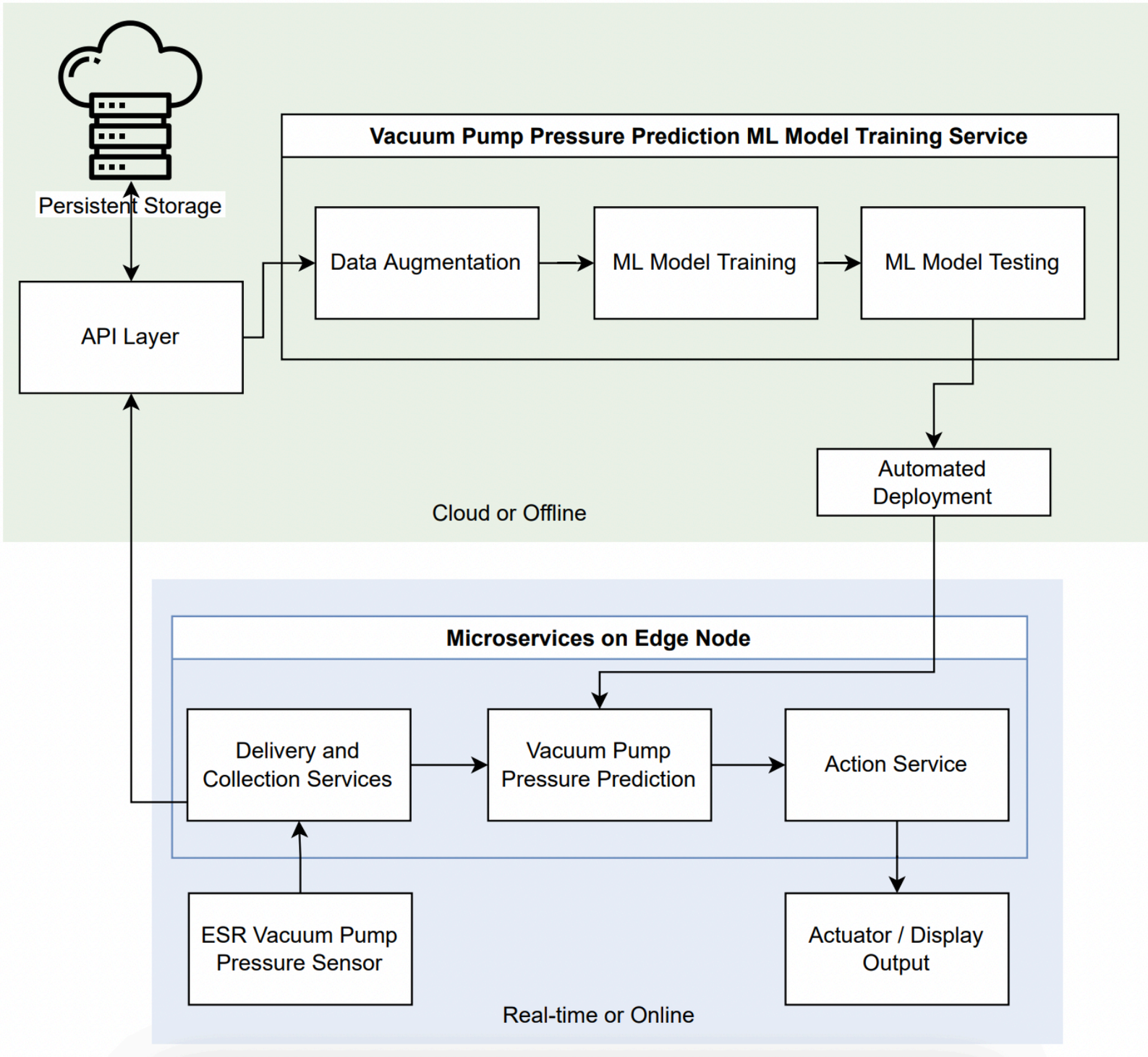


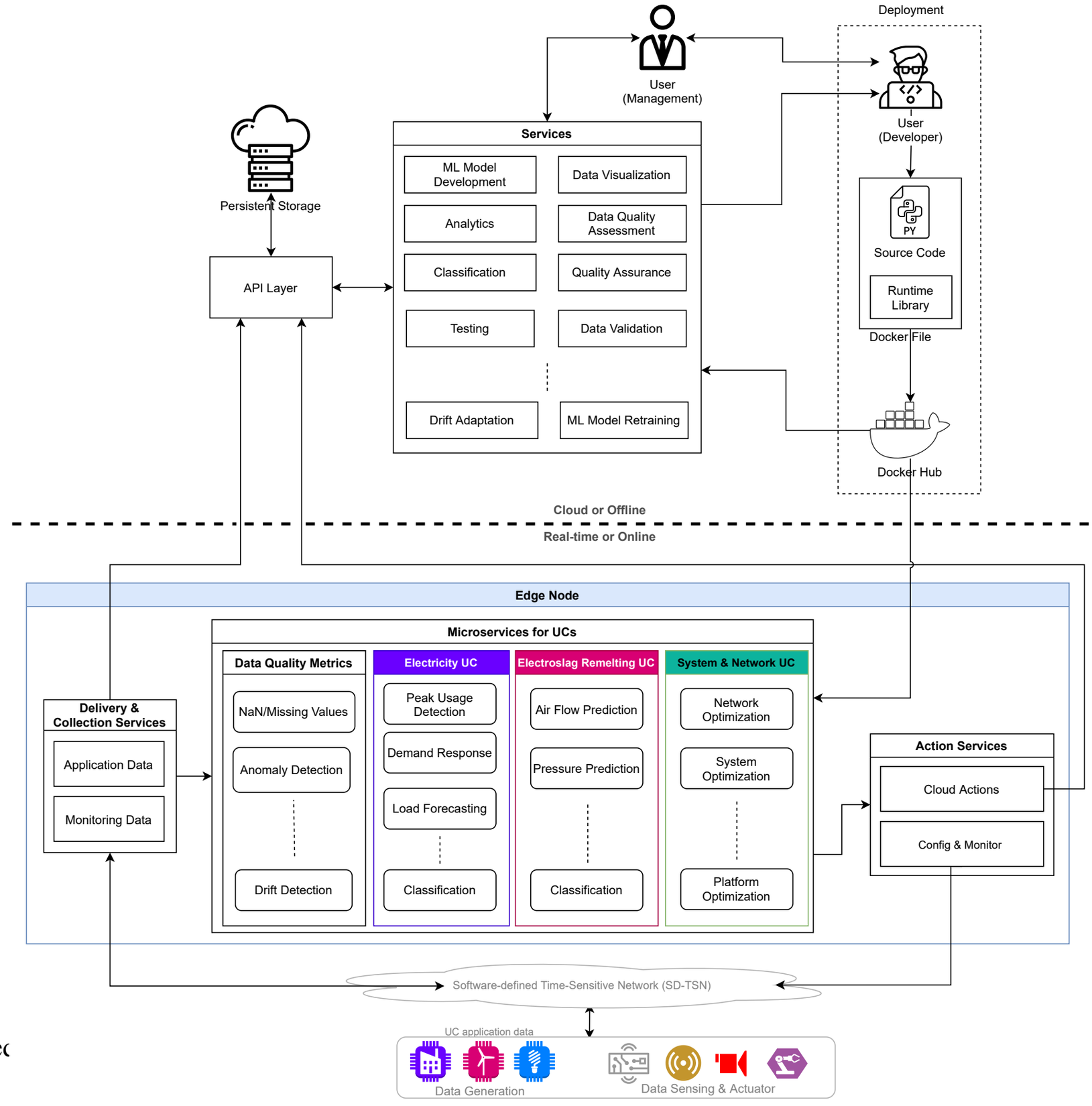
Image source: https://arxiv.org/pdf/2004.05785.pdf

Dealing With the Problems in MLOps and DataOps

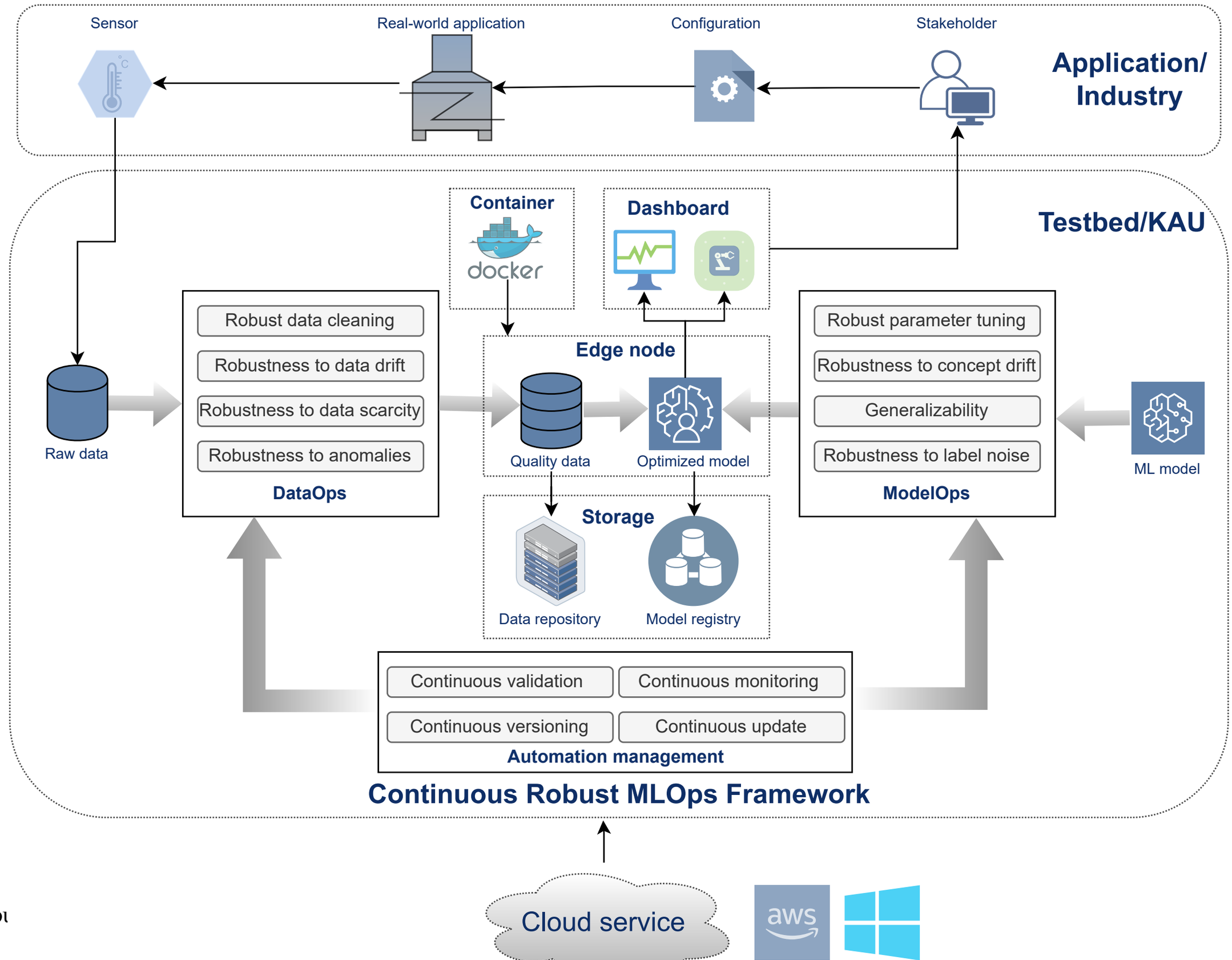
- Building customisable micro services to deal with the custom data and ML problems



End-to-End Approach

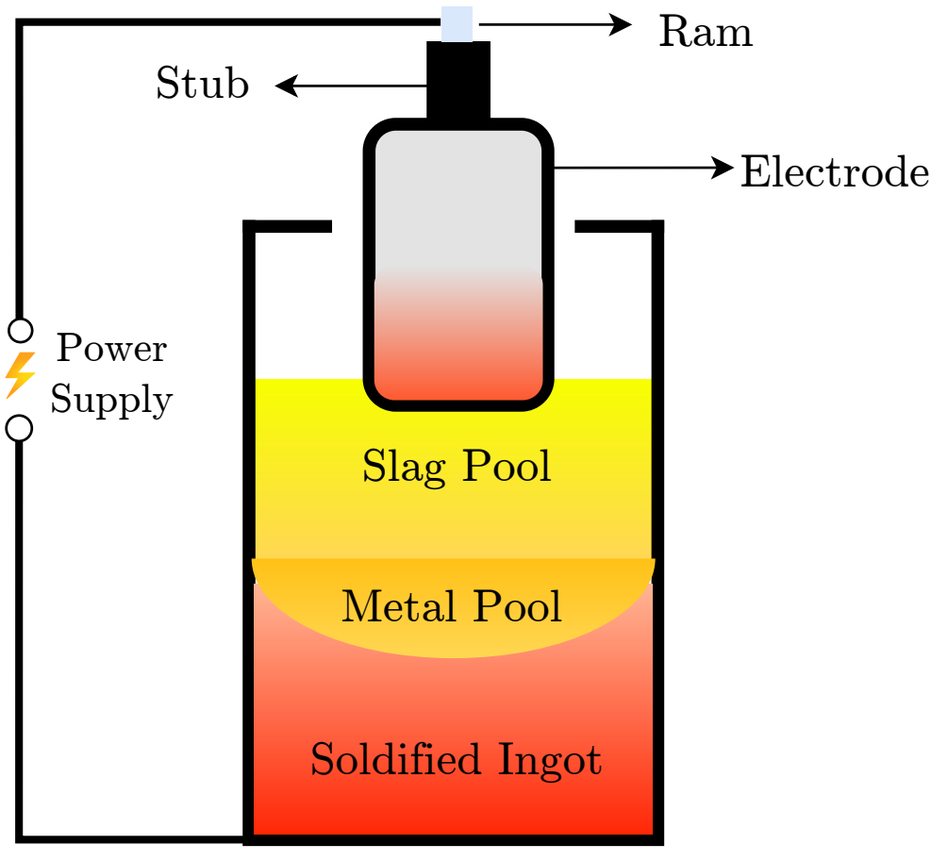


End to End Approach



Self-Adaptive Drift Handling

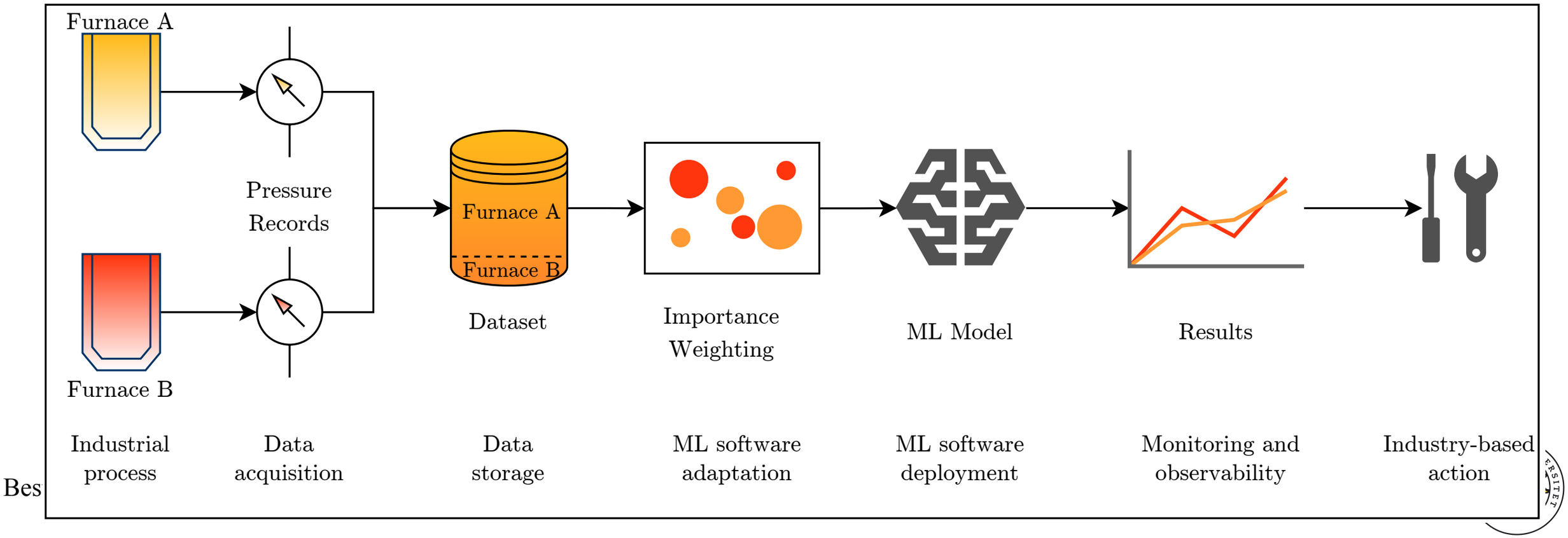
- Machine learning software to **predict the minimum pressure value** of a pumping event
- The minimum pressure value is predicted every **30 seconds** for up to **3 minutes**
- Evaluate: **Predicted value < pressure threshold**
- Benefit: **Early identification of invalid pumping events**



- Industrial process scalability: Introducing **a new furnace** to the industry
- Non-functional requirement: **Adaptability**
- **Fast integration** in the predictive system

Drift Handling Approach for Self-Adaptive ML Software in Scalable Industrial Processes

- **Collect** —> **Adapt** —> **Deploy** —> **Monitor** —> **Decision**
- Covariate shift adaptation: **importance weighting, Kernel Mean Matching (KMM)**
- ML model: **Random Forests(RF)** and **XGBoost**
- Evaluation metric: **mean absolute percentage error (MAPE)**

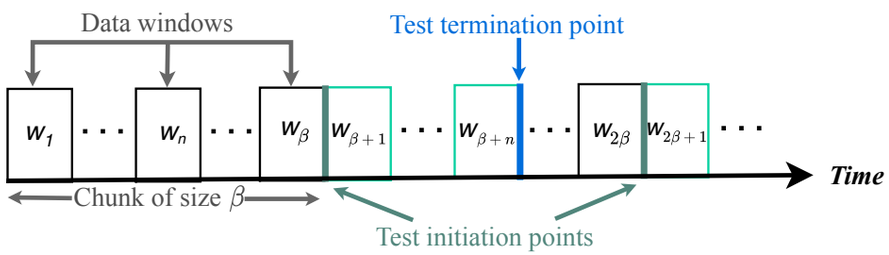


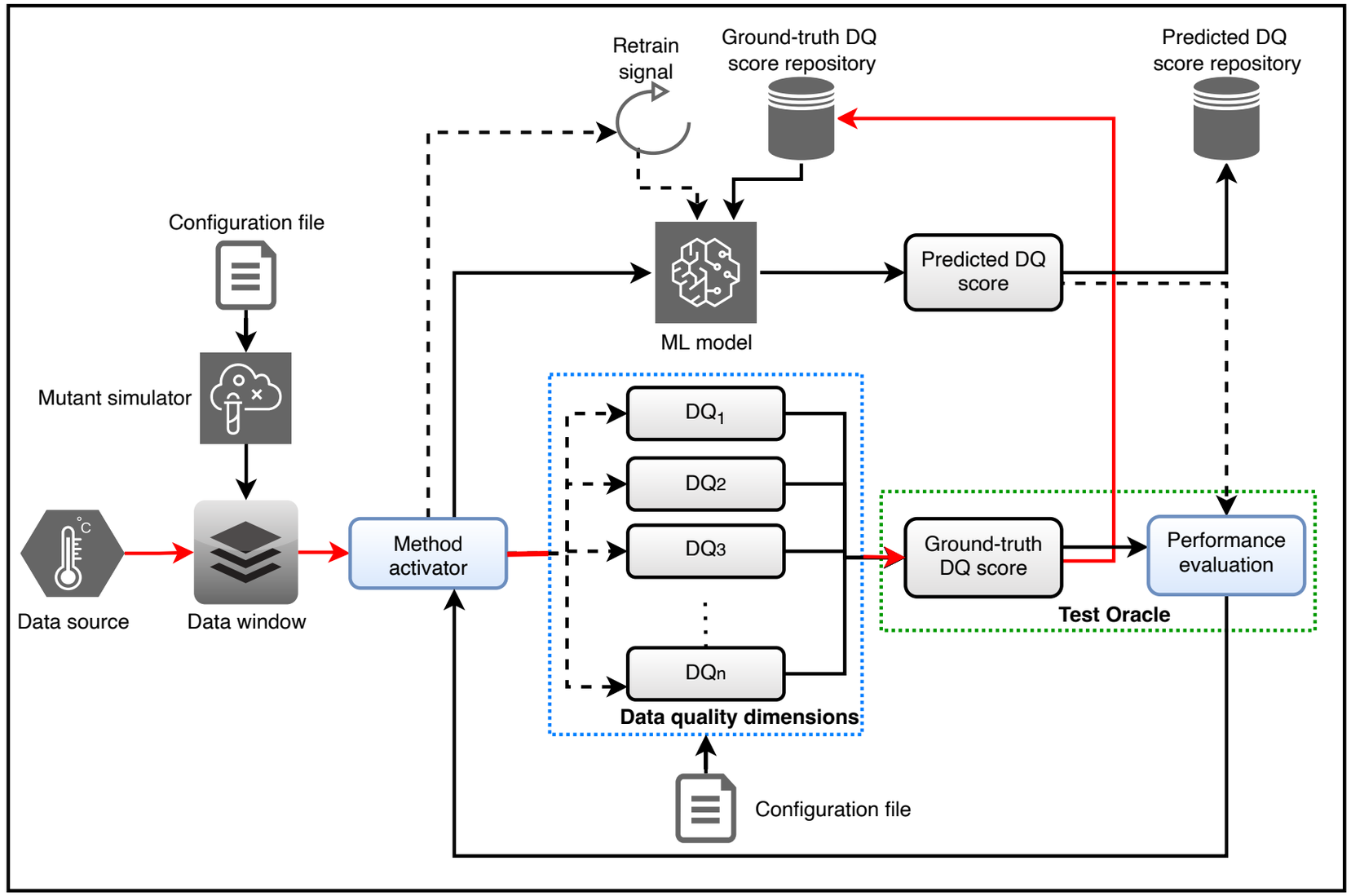
Bes

Data Quality Scoring

- ML approach
  - Score n data points using the pipeline approach
  - Train ML regression on the training datasets of size n
  - Predict the score of the testing dataset of size l

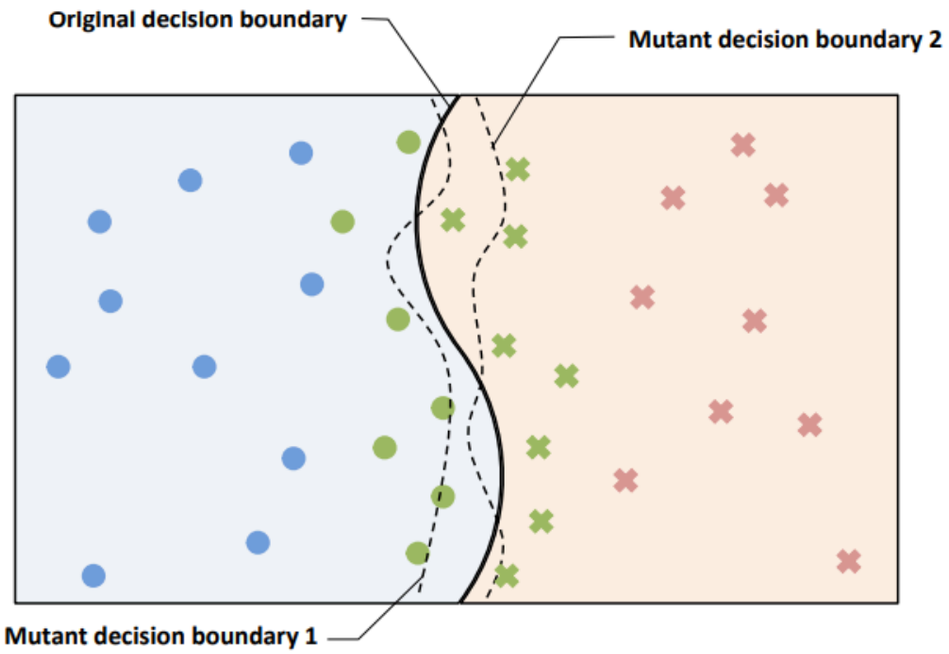
- **DQSOps: Continuous Data Quality Scoring Framework for Data-Driven Applications**





Mutation Testing and Data Quality Scoring

- **Mutation testing:** is a software testing technique that involves introducing small, deliberate changes (mutations) into the source code and testing the program to see if it detects and fails the mutations.

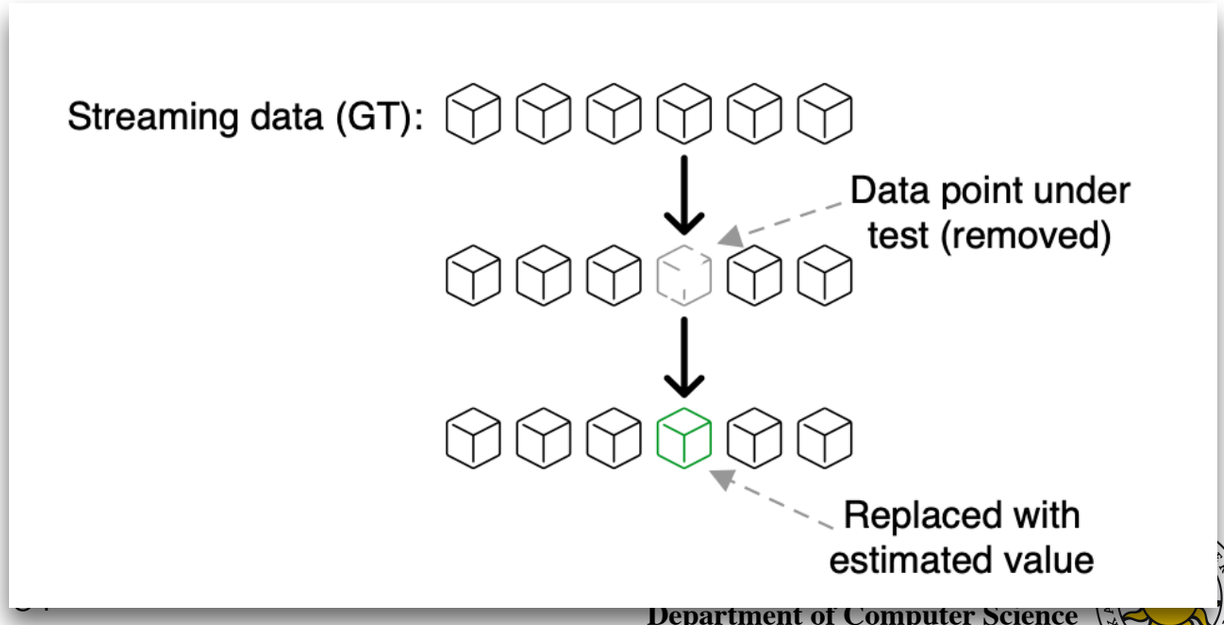


- Some attempts available in the literature by using new data points in the decision boundary.

- Our approach is to look at the data quality dimensions and play with the data:

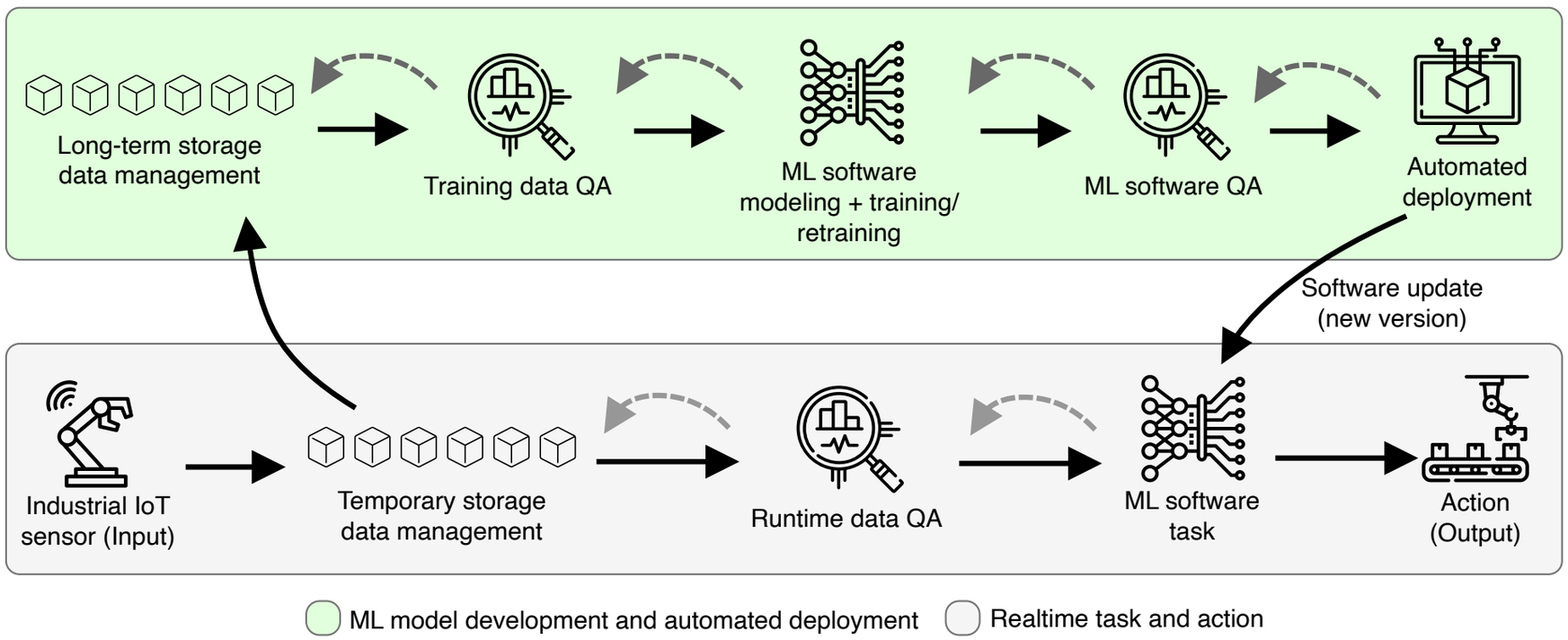
- **Intrinsic DQ:** How pure is the data?

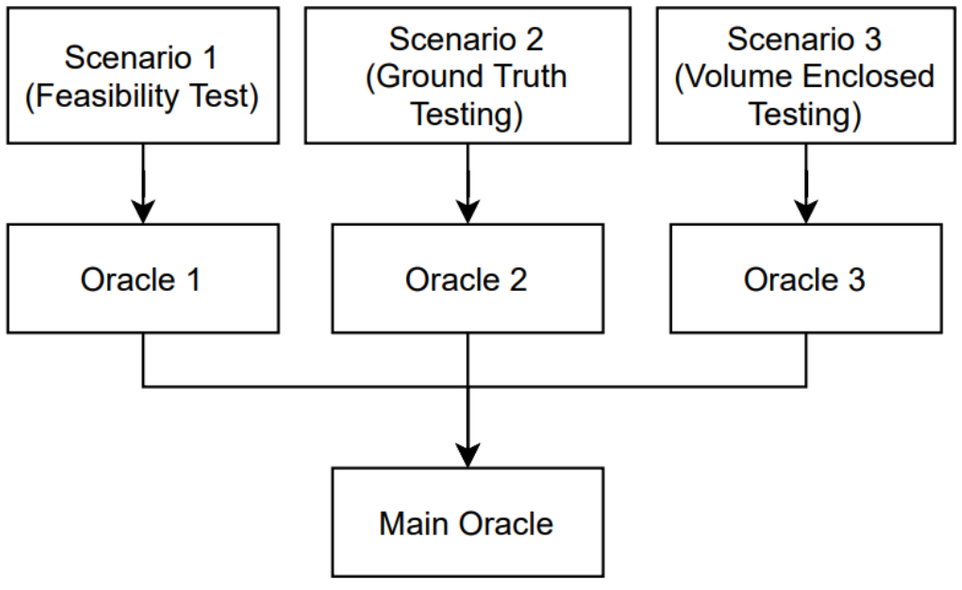
- **Contextual DQ:** How useful is the data for business applications?

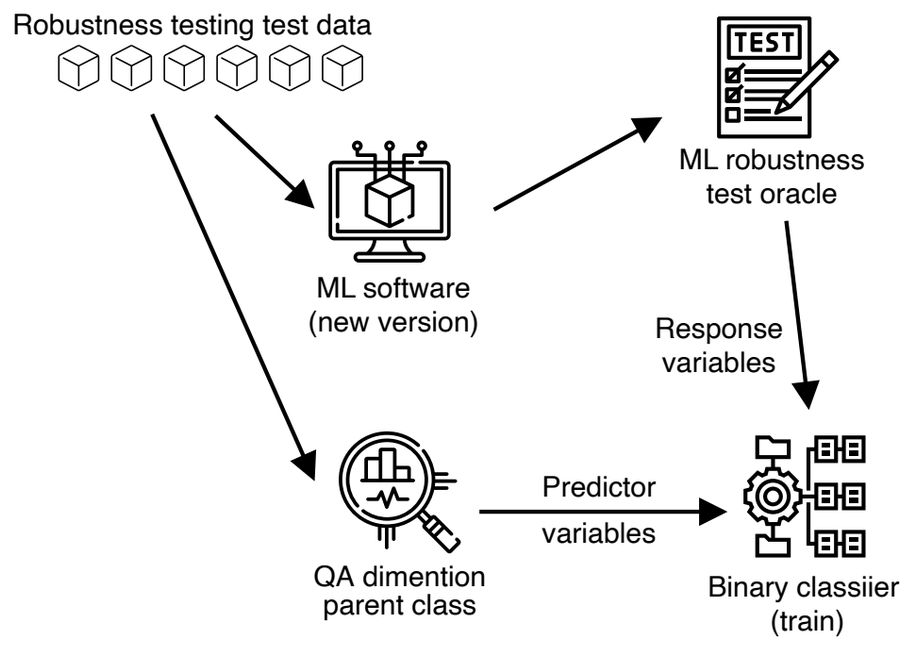


Reference: L. Ma et al., "DeepMutation: Mutation Testing of Deep Learning Systems," 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE), Memphis, TN, USA, 2018, pp.

Continuous Robustness Testing in MLOPs

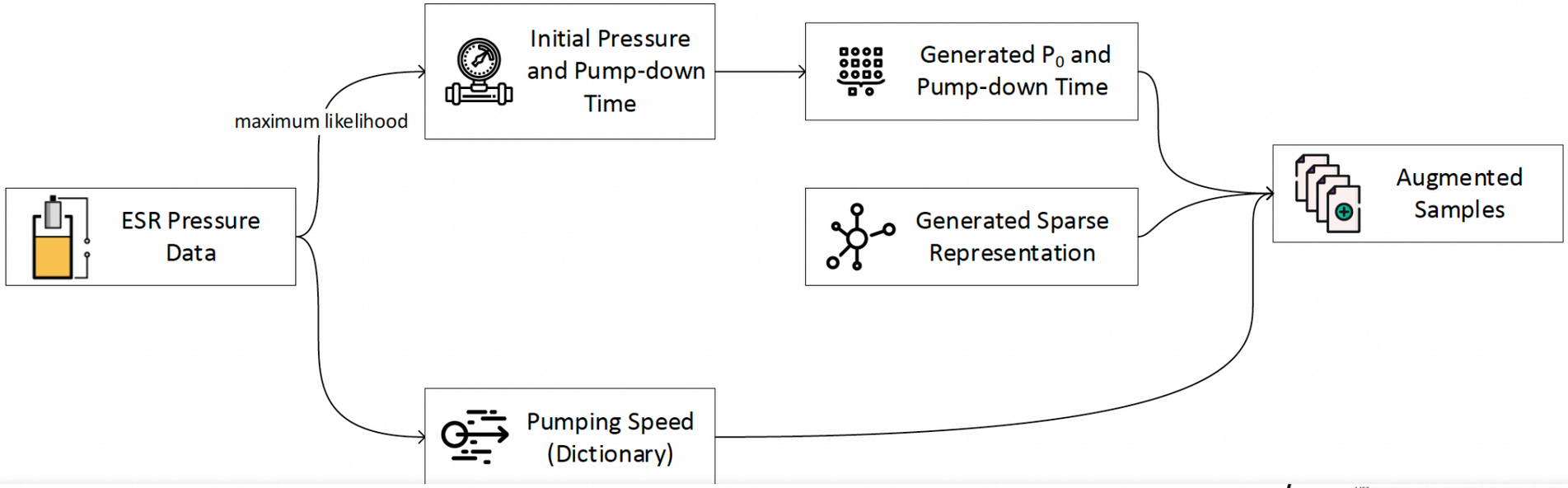


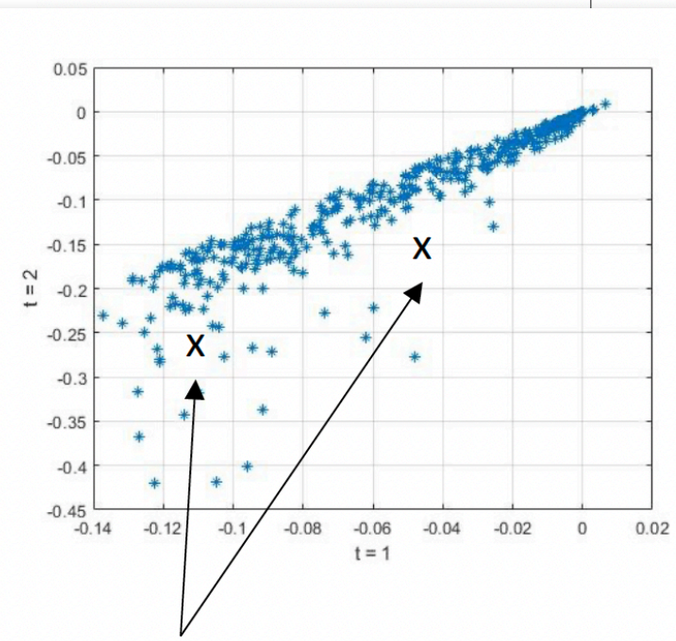




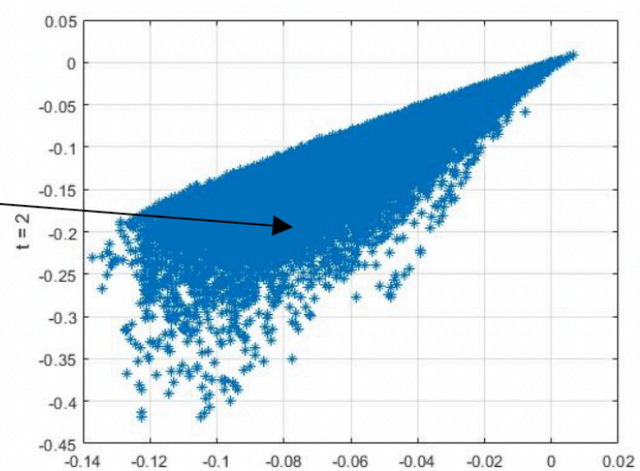
Data Augmentation for Limited Data

Continues creation of dataset

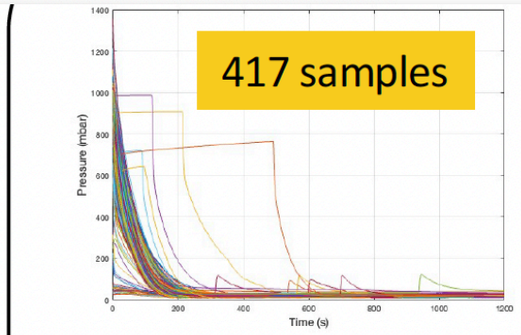


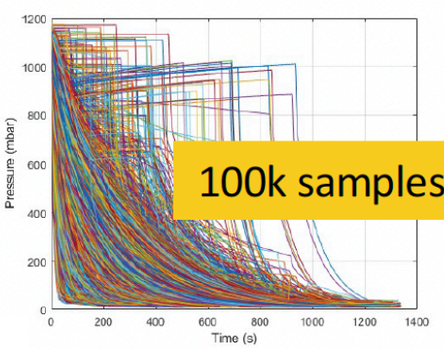


Gaps filled in augmented data

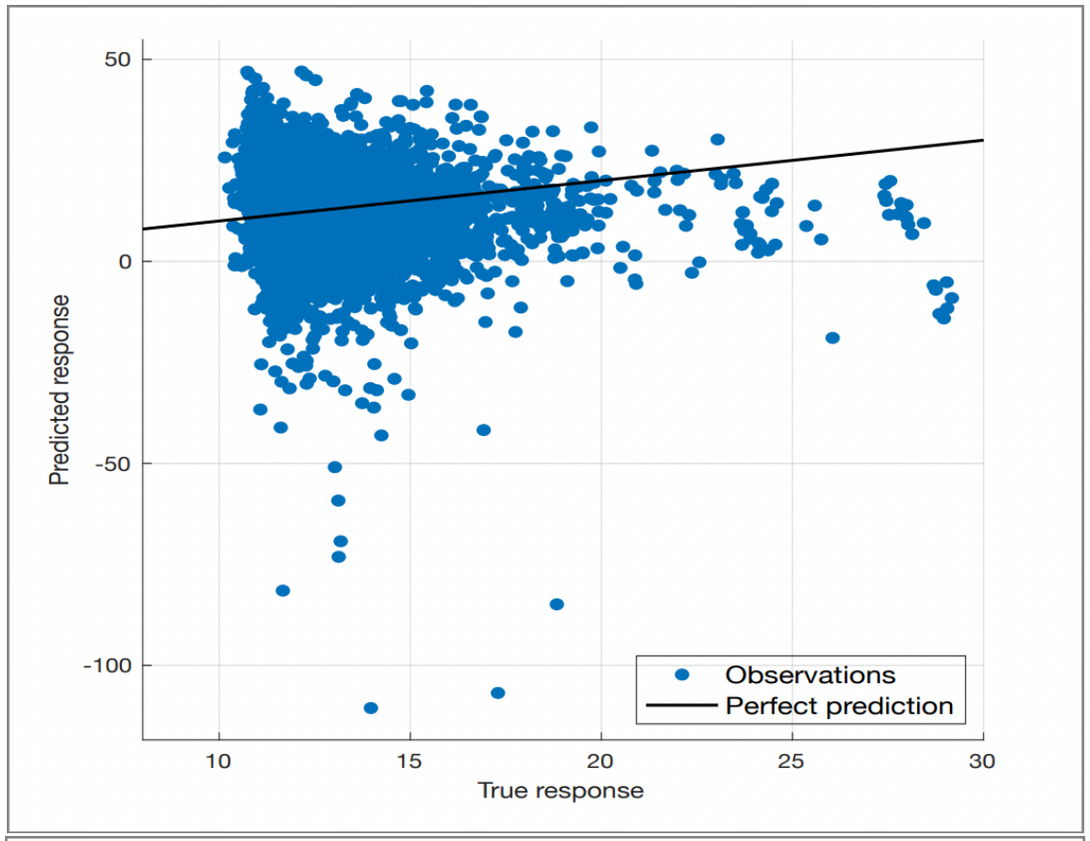


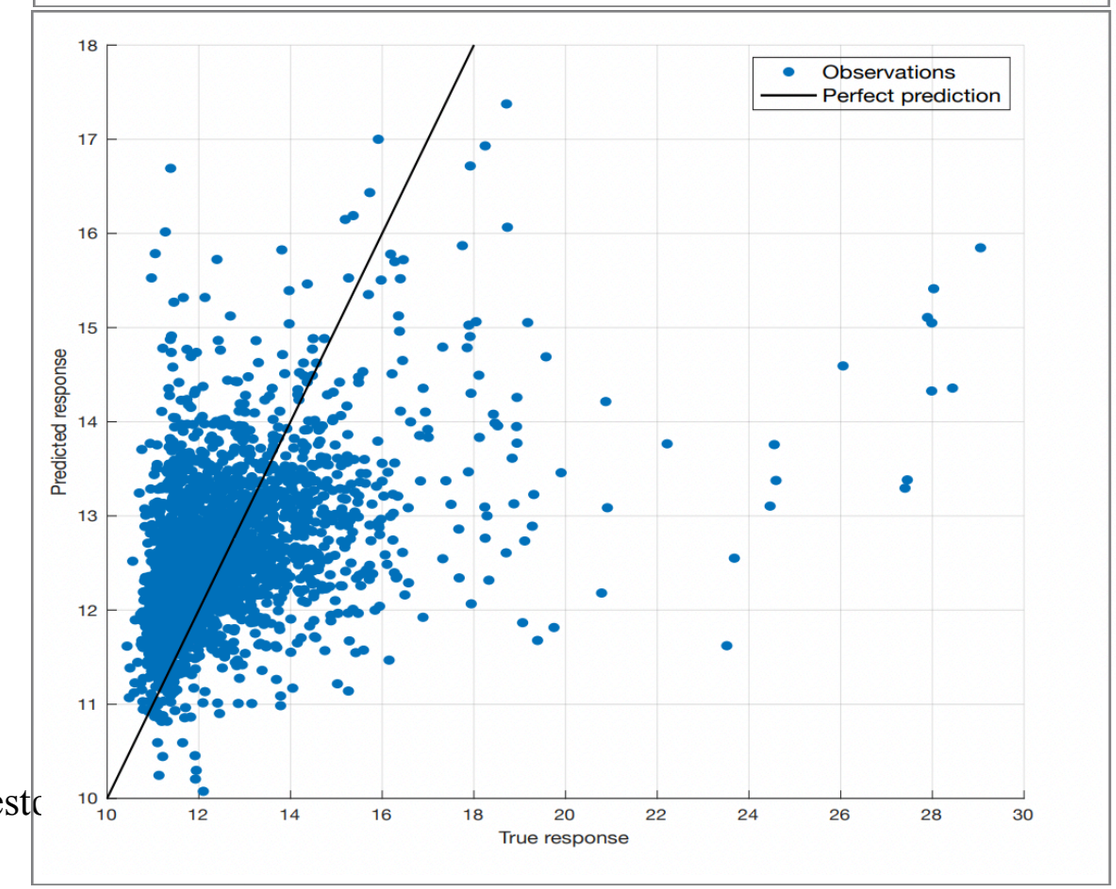
Pumping speed

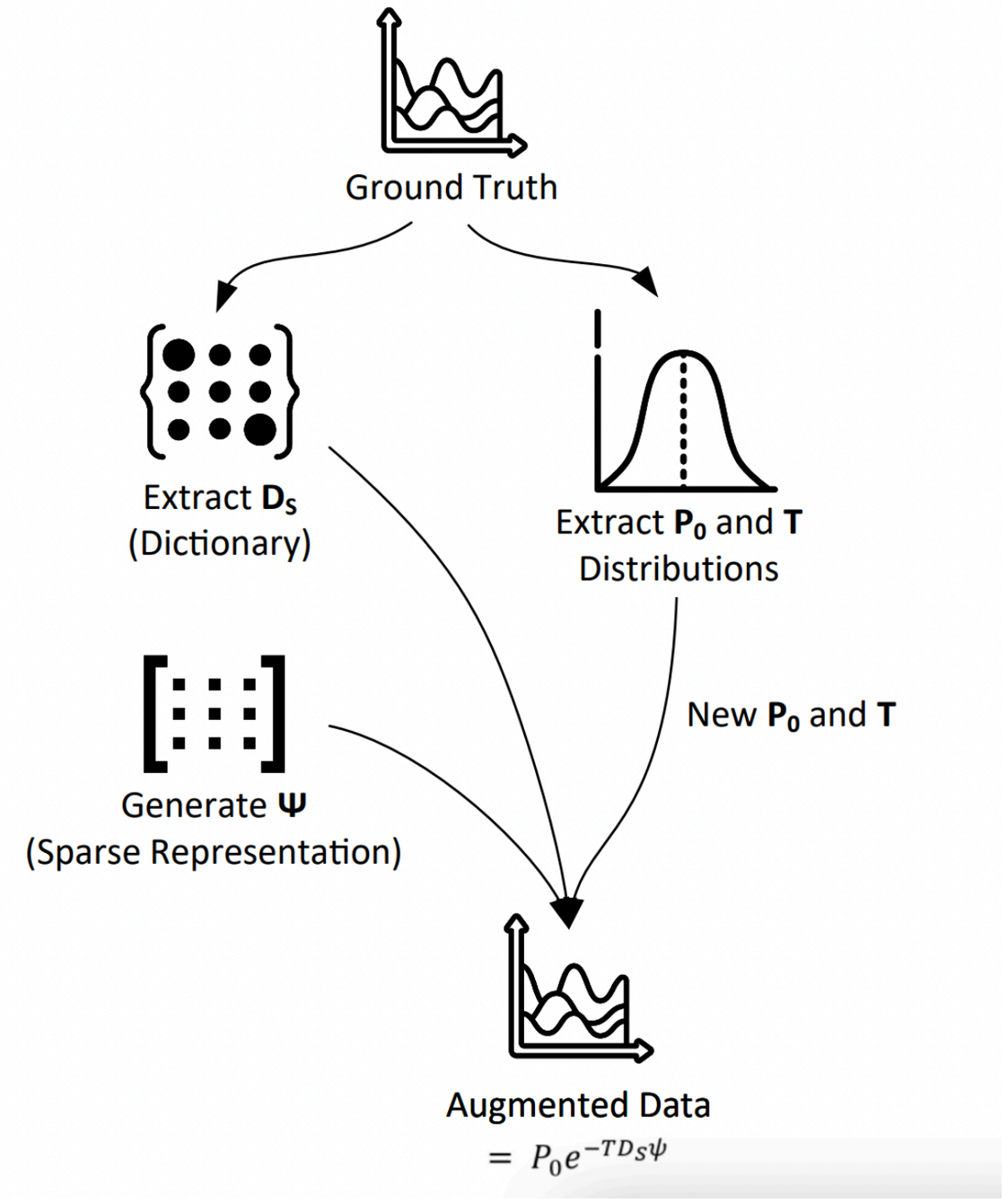




Using Data Augmentation for Robustness Testing



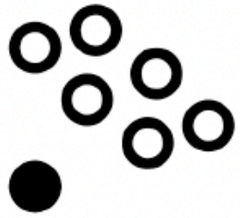




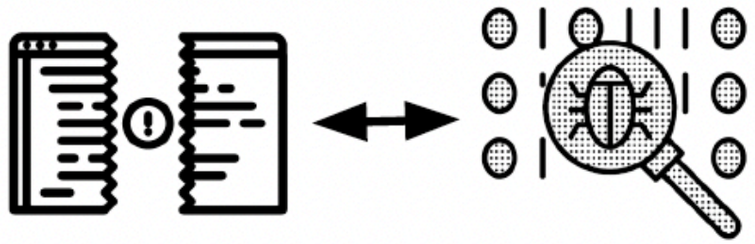
KARLSTAD UNIVERSITY
Department of Computer Science

Anomaly Detection in Customer Data

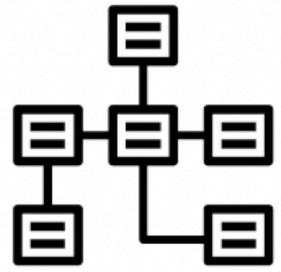
- Anomaly detection during data evolution



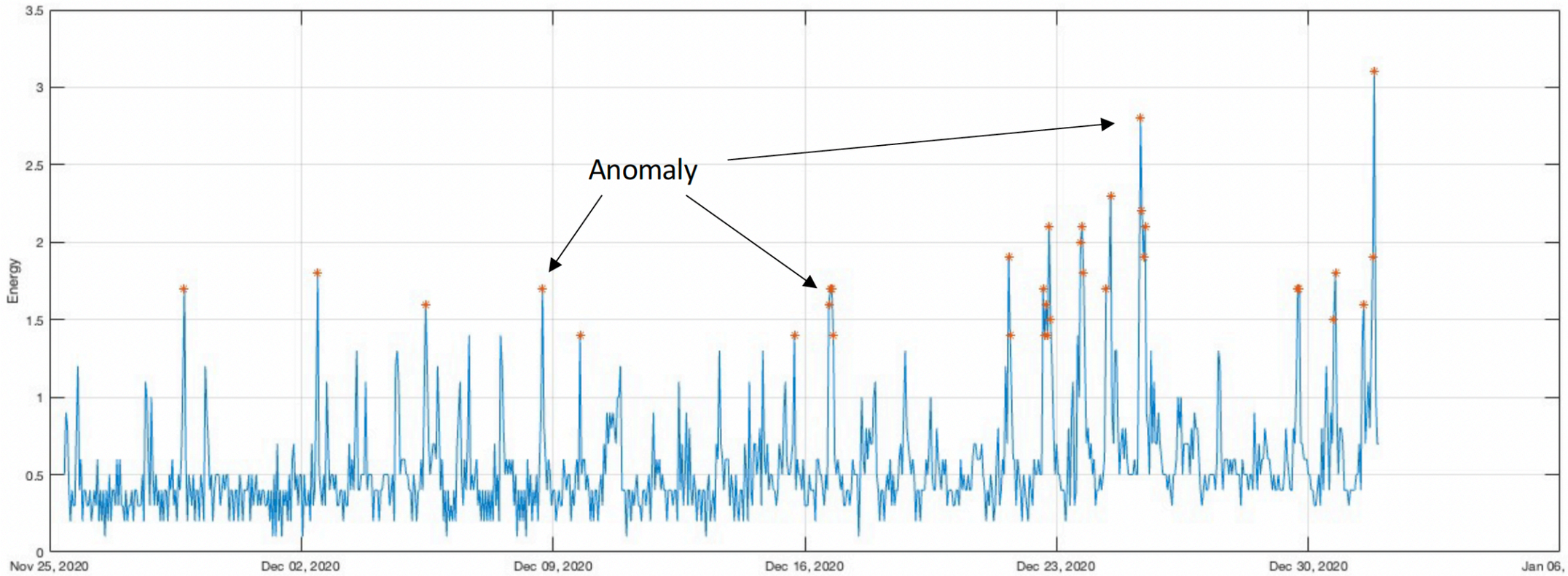
Easy to detect anomalies



Data errors treated same as code bugs



Update data schema

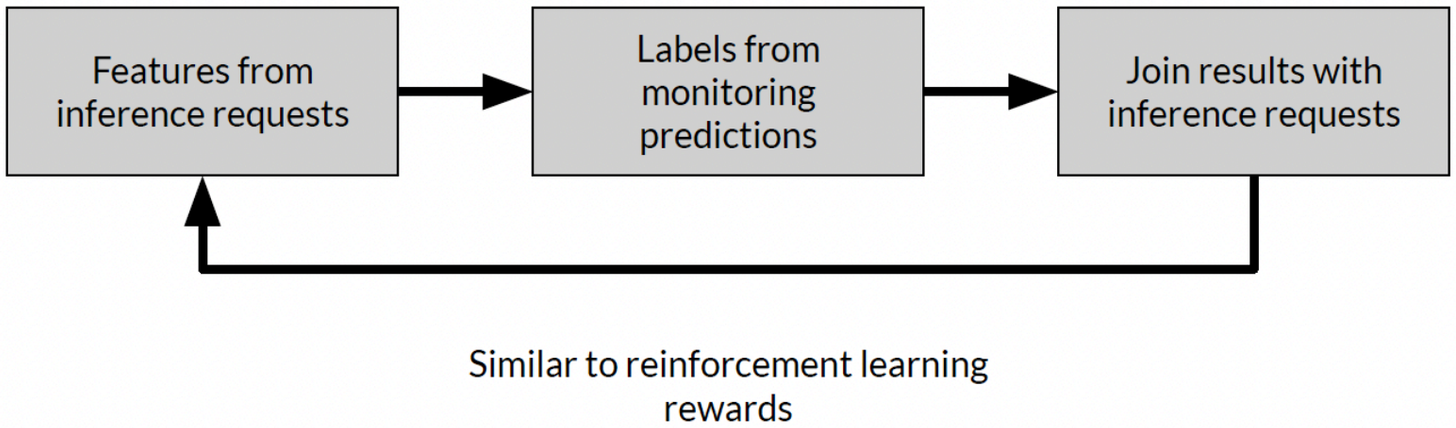


Bestoun

Source: Machine Learning Engineering for Production (MLOps), https://www.deeplearning.ai/program/machine-learning-engineering-for-production-mlops/

More Problems to Investigate

- Continuous creation and labelling of the dataset.



- Testing for model degradation and continuous mitigation in MLOps

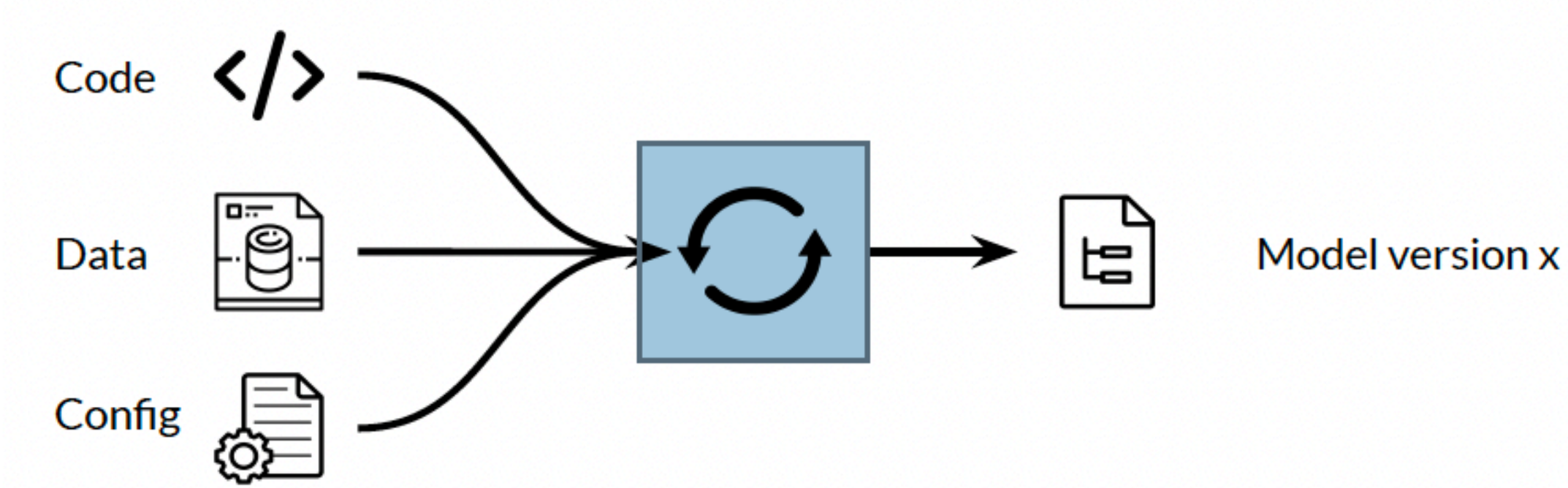
On-Demand	• Manually re-train the model
On a Schedule	• New labelled data is available at a daily, weekly or monthly basis
Availability of New Training Data	• New data available on ad-hoc basis, when it is collected and available in source database

Source: Machine Learning Engineering for Production (MLOps), https://www.deeplearning.ai/program/machine-learning-engineering-for-production-mlops/

Data and Model Versioning and Preservation

- Experiment tracking to compare results and how to use for testing

Name (50 visualized)	Tags	acc	Sweep	optimizer	epoch	batch_size	n_train	n_valid	n_conv_lay	loss	GPU
batch 64 4 GPU	4GPU b_64_c	0.4305	-	rmsprop	49	64	5000	800	1	1.632	-
batch 64 (V2, 5K train	2GPU b_64_c	0.4343	-	rmsprop	49	64	5000	800	1	1.63	-
50K examples (b 64)		0.4042	-	rmsprop	49	64	50000	8000	1	1.76	-
batch 32 4 GPU	4GPU b_32_c	0.4032	-	rmsprop	49	32	5000	800	1	1.714	-
batch 64 1 GPU	1GPU GCP	0.4465	-	rmsprop	49	64	5000	800	5	1.615	1
batch 128 (5K train)	2GPU b_128	0.4181	-	rmsprop	49	128	5000	800	1	1.658	-
batch 256 4 GPU	4GPU keras	0.3882	-	rmsprop	49	256	5000	800	1	1.751	-



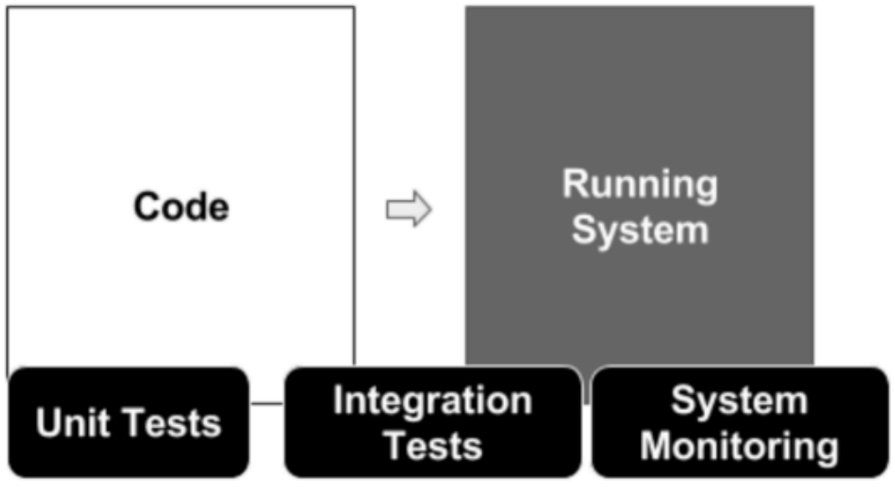
Bestoun

Source: Machine Learning Engineering for Production (MLOps), https://www.deeplearning.ai/program/machine-learning-engineering-for-production-mlops/

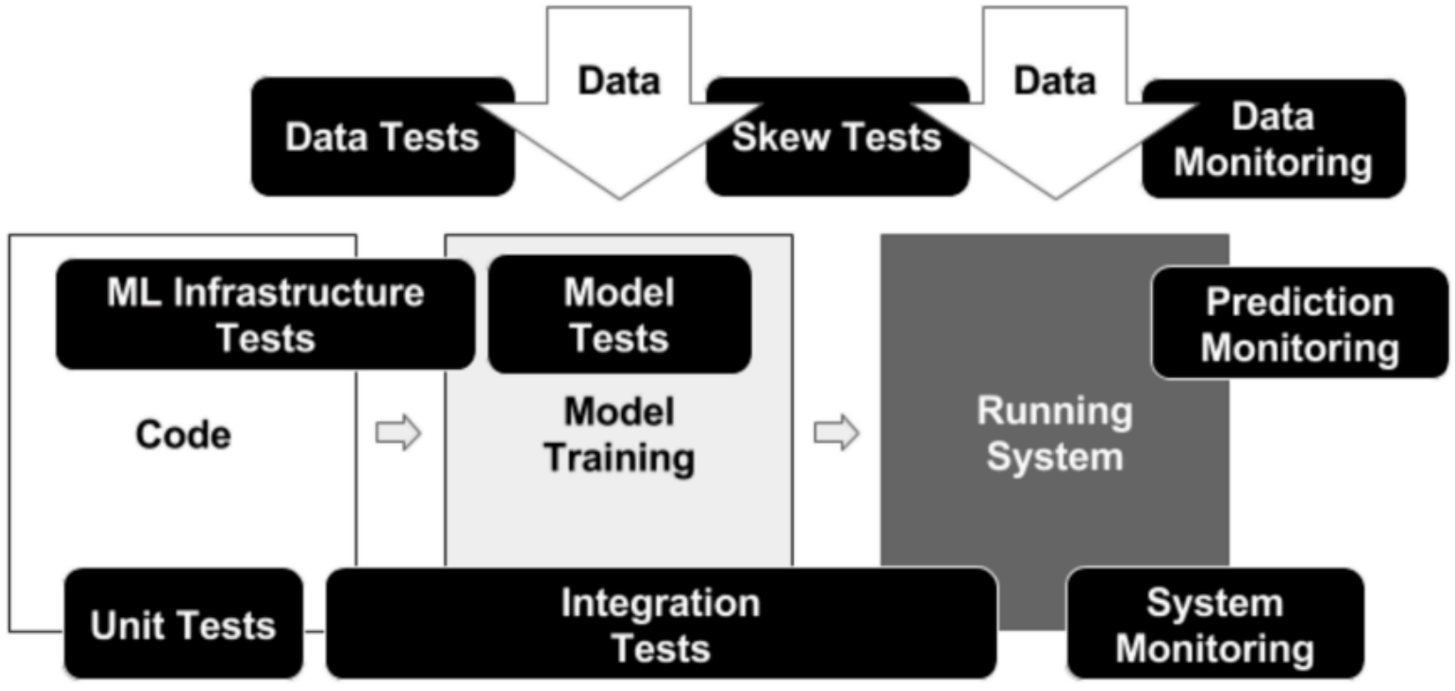
Testing in an MLOps system is different

- Testing ML applications is, inherently more complex than traditional systems.

- **In addition to traditional software testing:**
  - Data tests
  - Model tests
  - Pipeline tests



Traditional System Testing and Monitoring

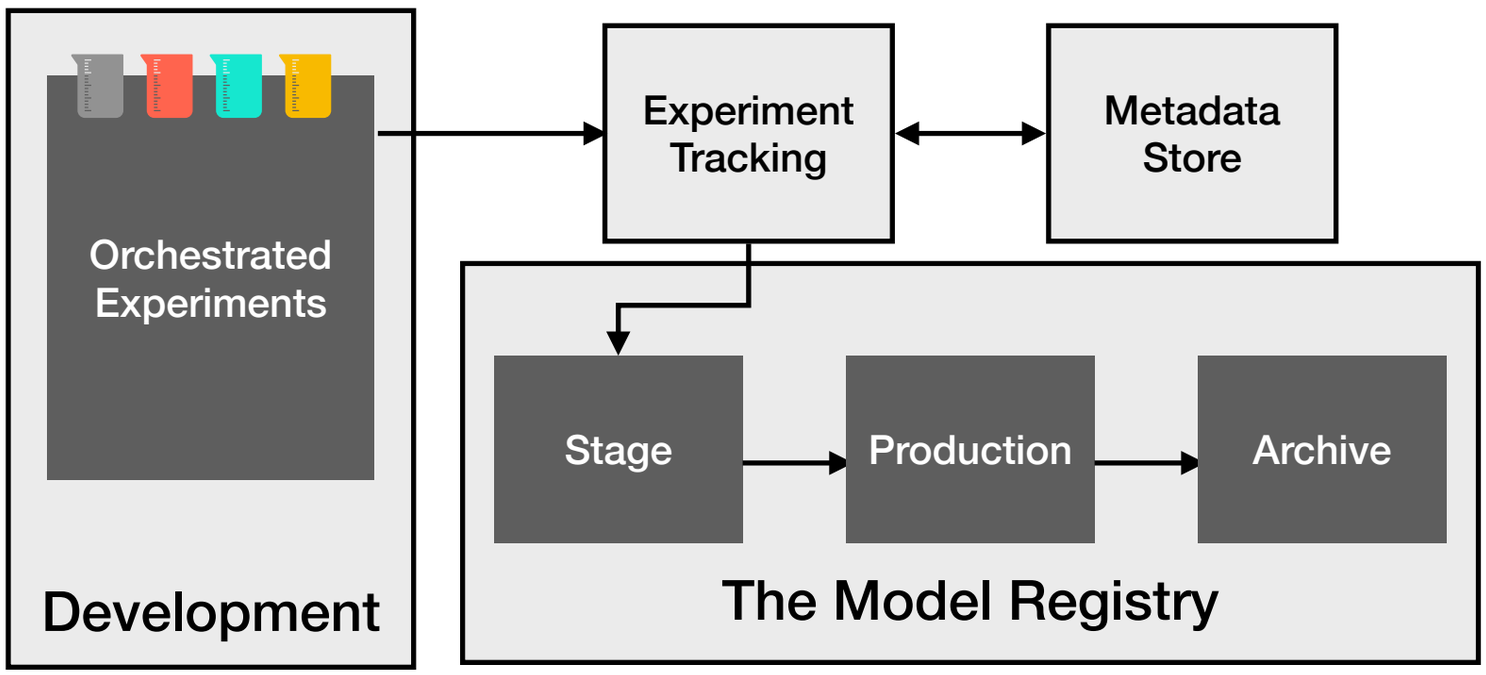


ML-Based System Testing and Monitoring

Model Registry? Experimentation

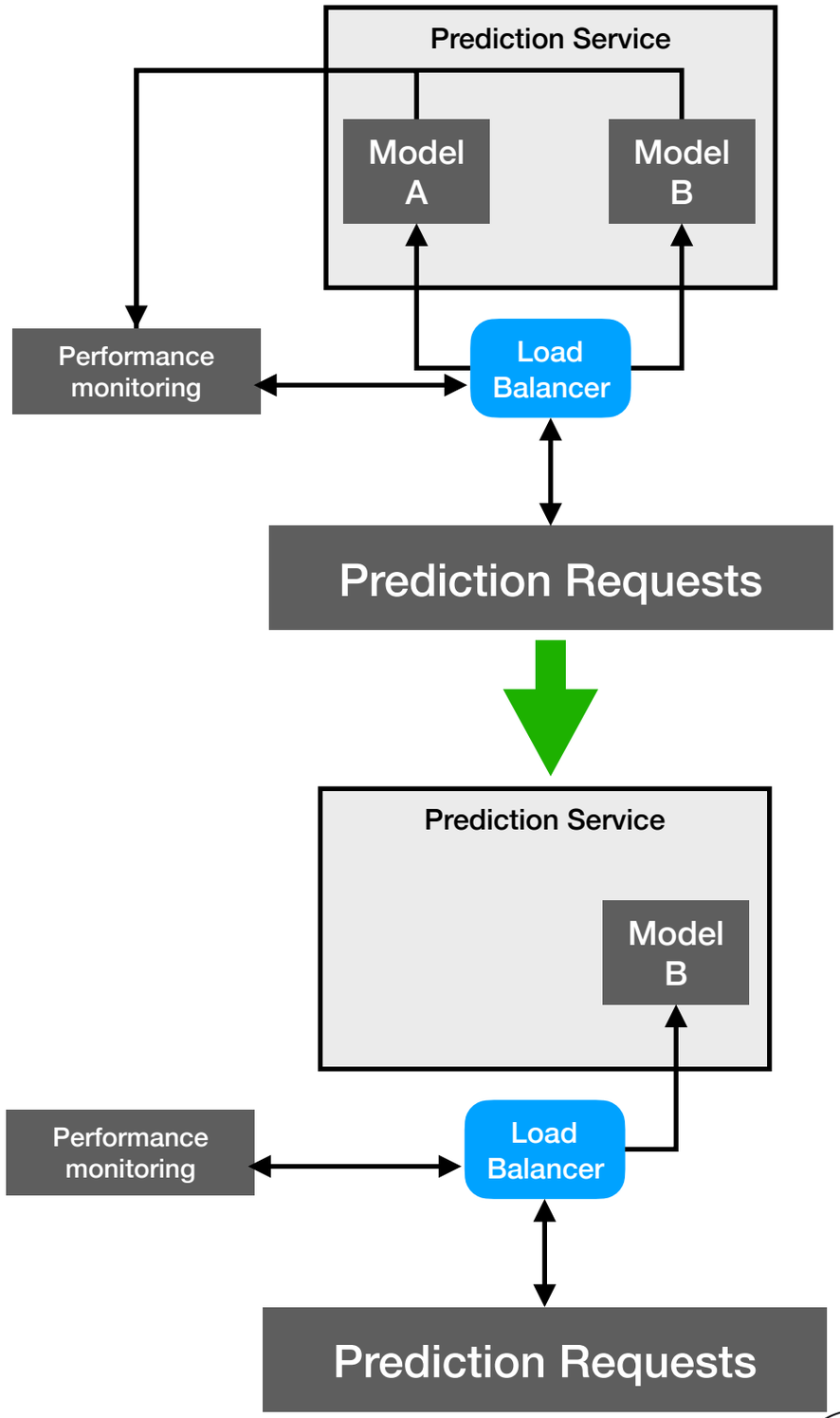
- The model registry plays a central role in managing the lifecycle of ML models. It serves two primary functions:
  - **Centralized Storage:** The model registry acts as a centralized repository for storing models and all their associated artifacts such as the trained model files, metadata, configurations, and any other resources required for deployment.
  - **Collaborative Lifecycle Management:** Beyond storage, the model registry also facilitates collaborative management of the model's lifecycle. It enables teams to track the progression of models from development to deployment and beyond. This involves versioning, documentation, and tracking changes made to the model.

- In the experimentation, we can select the model with the best performance.



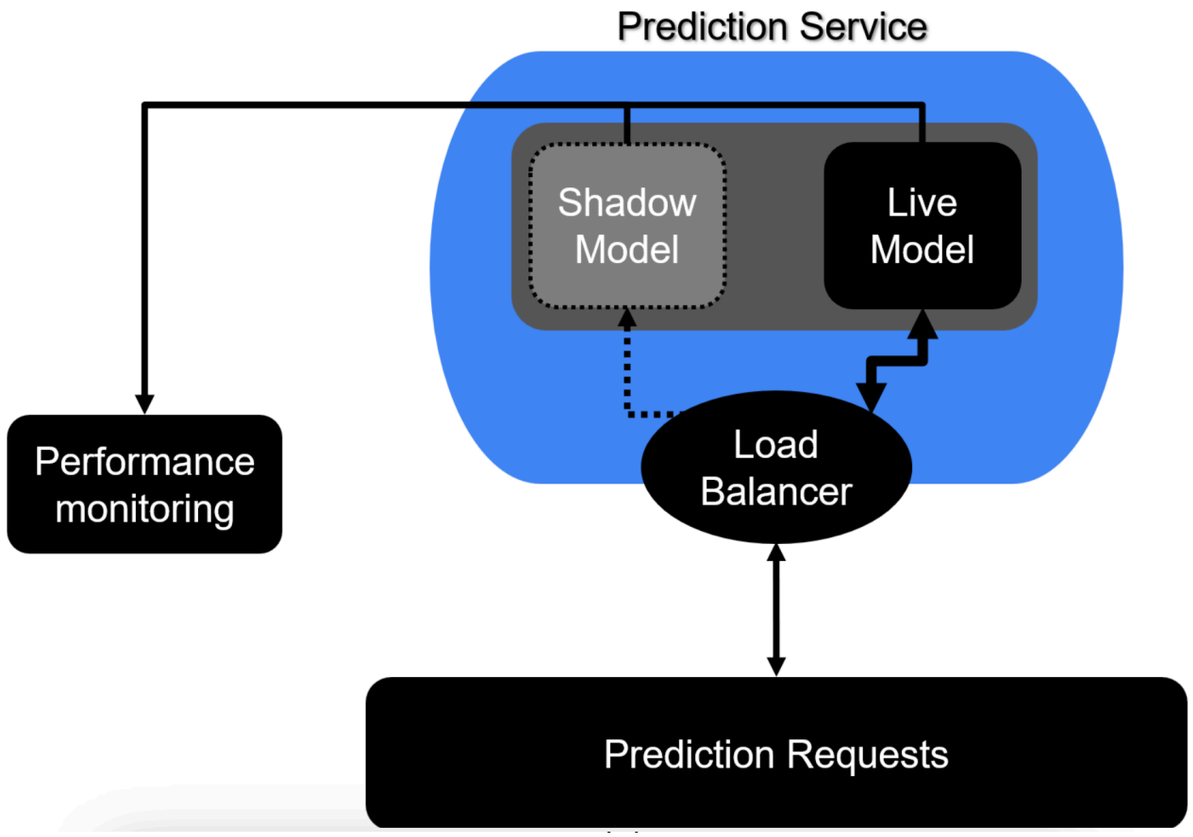
A/B testing

- There are two models, A and B, running in parallel.
- A load balancer directs client requests to these models.
- Continuous performance monitoring helps the load balancer adjust request distribution between the models.
- The aim is to route most requests to the better-performing model, leaving the other for validation.
- By automatically shifting requests to the better-performing model over time, an A/B testing deployment strategy allows the deployment to automatically adjust and use the best-performing model, ensuring the highest quality of predictions.



Shadow Deployment

- A new model runs alongside the production model.
- Requests are divided between them, but only the live model provides predictions to clients.
- Both models' outputs are compared and monitored for performance differences.
- If the shadow model outperforms the live one, it can replace it as the new production model.

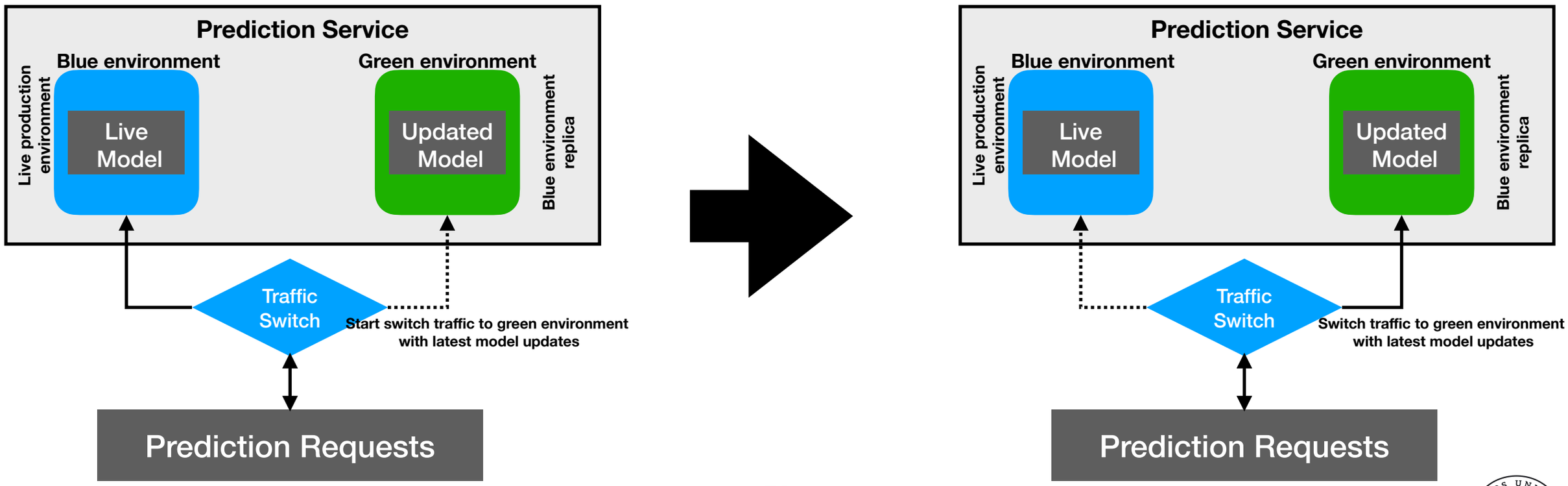


Blue/Green Deployment

- The "blue-green" deployment strategy involves transitioning from a model running in production in a "blue" environment to replacing it with an updated model in a "green" environment.

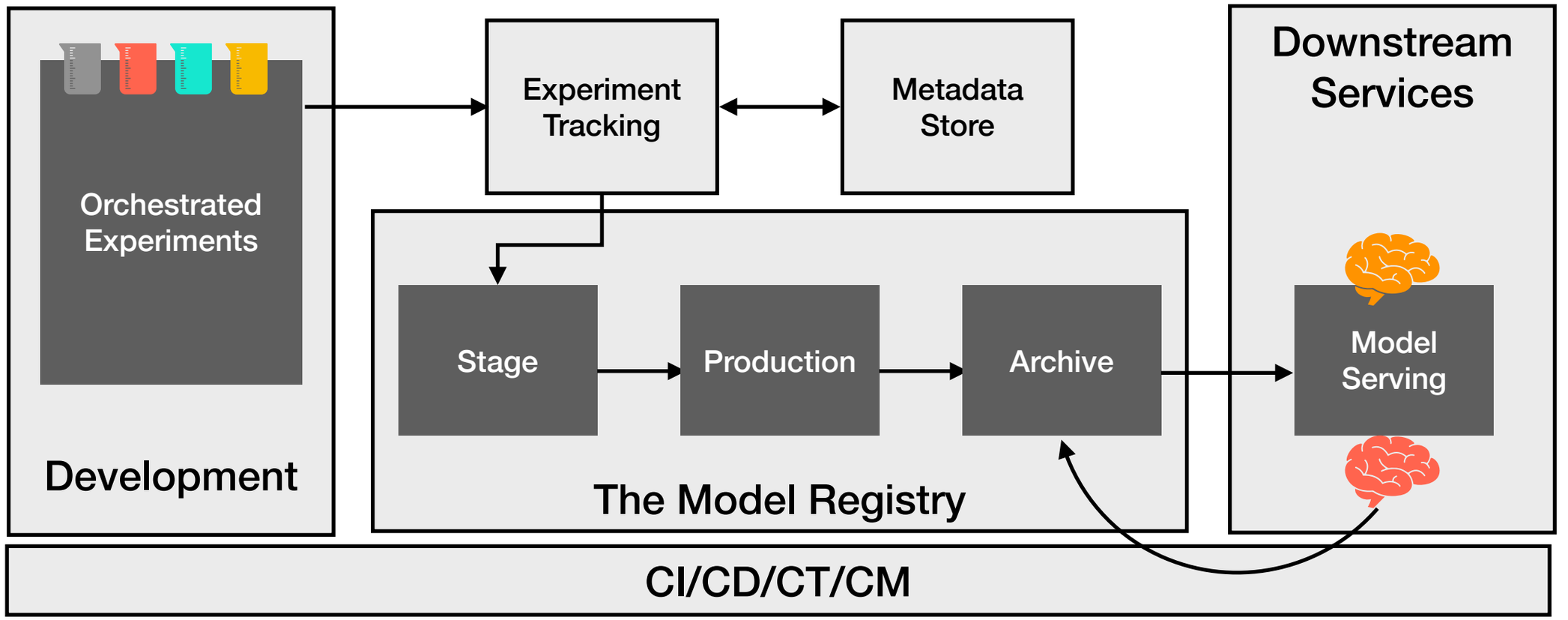
- This transition occurs gradually, following specified criteria, as prediction requests switch from the blue to the green environment.

- If the system functions without problems, most traffic eventually shifts to the green environment, until finally, all traffic is handled by the updated system in the green environment.

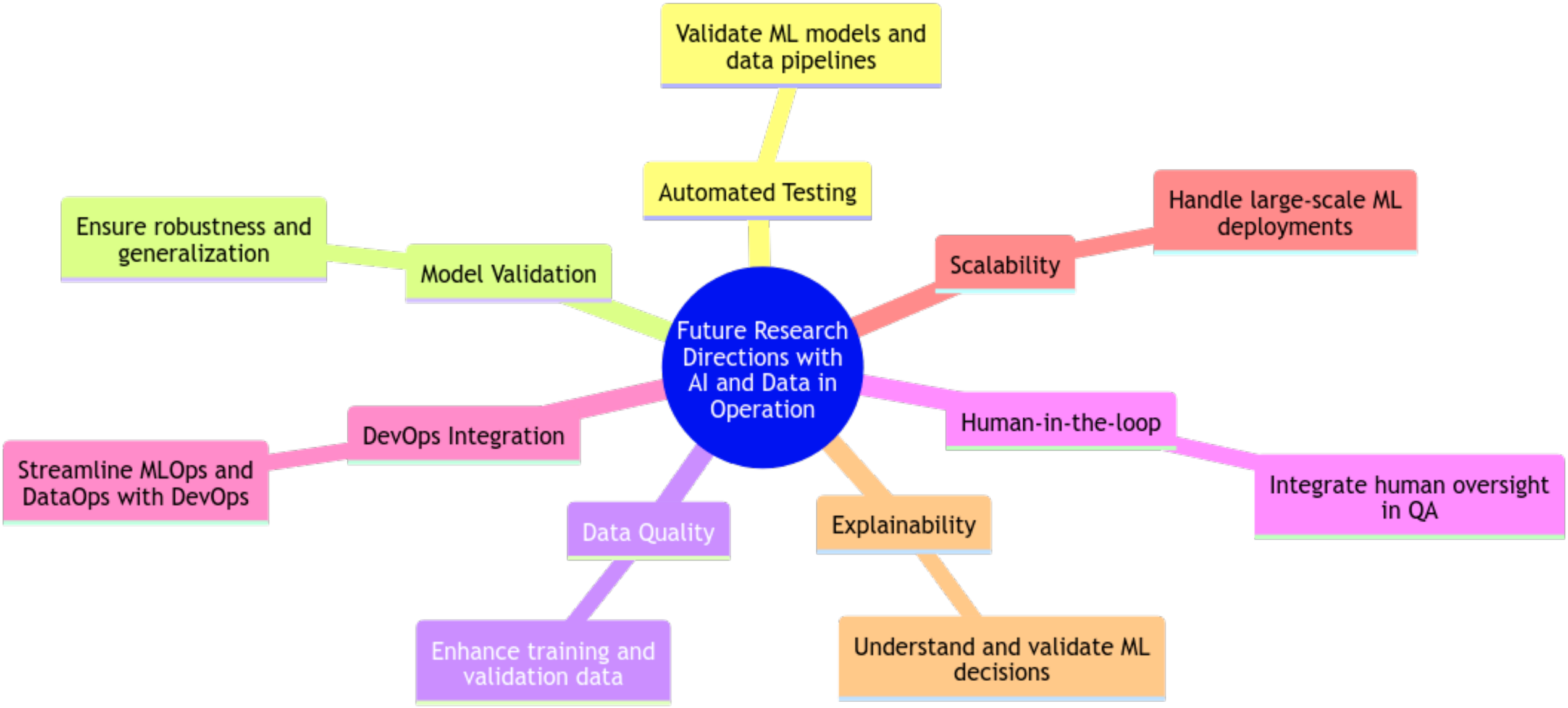


Model Registry? Model Decommission

- Finally, the model that was in operation before the registration of the newly deployed model will be decommissioned and archived in the model registry.



Future Research Directions and Challenges




KARLSTADS UNIVERSITET