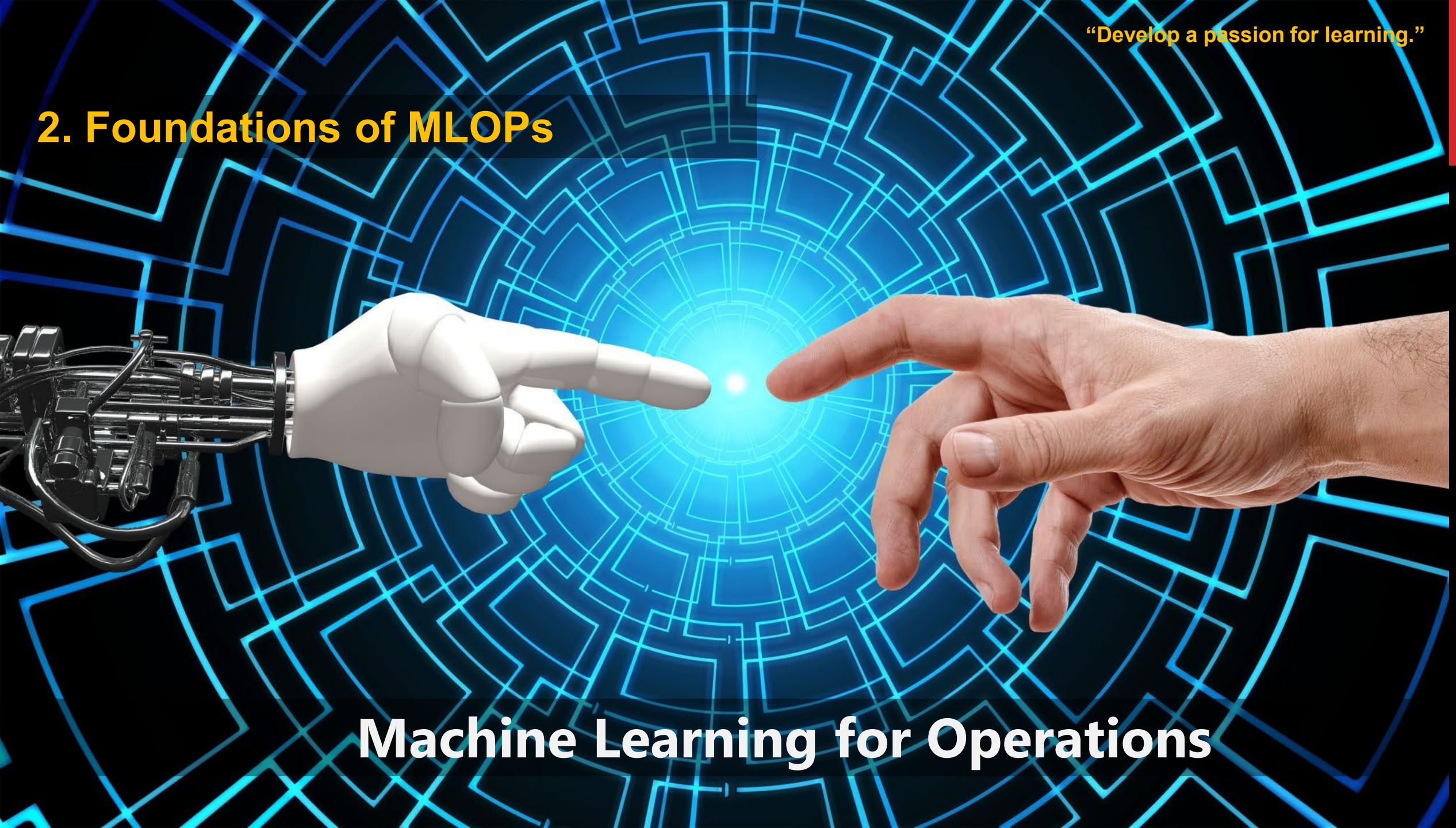
2. Foundations of MLOPs



Machine Learning for Operations

A Primer on Machine Learning

Operations Perspective on ML

- Applications that host machine learning models are different from most other applications.
  - For example, the model development process is based on complex mathematics and involves experimentation over many iterations.
  - In addition, applications that expose trained models might have different hosting requirements and strategies than standard applications.
- Trained models are sensitive to change in data and (potentially) business rules
  - A model-based application that works well when first implemented might not perform as well days, weeks, or months after being implemented.
  - For example, product recommendation models that are impacted by changing market conditions, such as seasonality of data or fraud detection models, might not be able to respond to new fraudulent practices.
  - To account for these differences, you need different processes and procedures for applications based in managing machine learning (ML).

Putting ML Apps into Production

- Why do ML Apps not make it to production?
  - The focus when designing and implementing an ML workload is on getting the model to work in a development environment.
    - There is often less focus on operating the model and making it accessible and reliable for users once it is trained.
  - ML might be a new IT discipline for the organization.
- Even when ML becomes operational, they might not stay operational.
  - Issues frequently turning into emergencies is an indicator that the ML app might be missing ML operations components.
  - The deployment is a patchwork of quick fixes and patches
  - Inconsistencies in the deployment process means not being able to react quickly enough to changing conditions within the operational enviroment

Possible Pain Point Questions

- Are there repeatable processes, or are issues resolved as unique one-time instances?
- Does the ML solution no longer meet the business purpose?
- Is there model decay, in which the predictive value of the model diminishes over time?
- Do the dev and ops teams coordinate well together?
- Does the model worked well during pilots, but fails in production?
- Is there an accurate way to track or measure the accuracy of predictions?
- Is the code difficult to understand, change and modify?

Idealized Team Roles

- Business stakeholders - business requirements
- Data engineers - locate, process, and prepare data for use.
- Data scientist - generate business insights and useful inference from the data.
- Machine learning engineer- design self-running software to automate predictive models.
  - The data scientist and the machine learning engineer both work with lots of information and have model-building skills.
  - The data scientist is focused on insights.
  - The machine learning engineer is focused on automation and on building models.

Idealized Team Roles

- DevOps engineer - responsible for the ongoing operation and maintenance of systems
  - MLOps engineer addresses the unique challenges of operating and maintaining systems that use machine learning models and inference.
- Software engineer - integrate inference into the application software, typically writing code that uses the inference API.

Case of the Murder Hornets

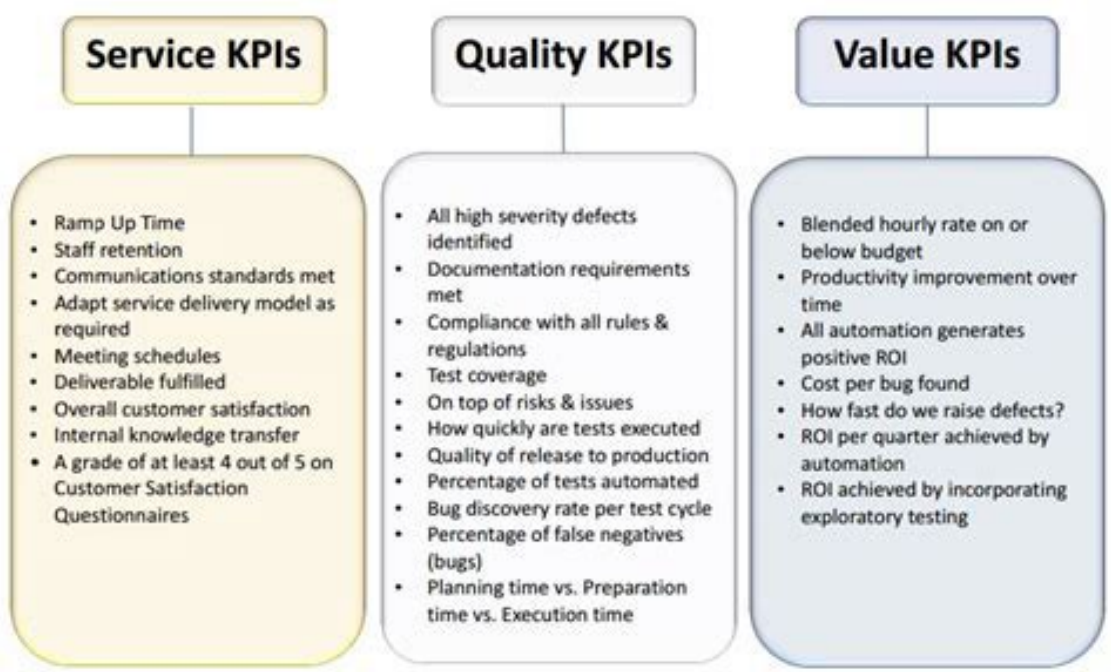
- Agricultural extension agents use an application to identify insects
  - The application is not identifying this insect. -
  - The image of the insect was sent to the county entomologist who identified it as a new species in the region: a murder hornet.
  - The murder hornet is capable of infesting a beehive and quickly reducing the bee population.
  - It is important that the agricultural extension app recognize murder hornets, but they were not even known in the region when the ML model was trained.

Case of the Unhealthy Health Checker

- A new feature of a health food company is a lifestyle health checker ML app that makes purchase recommendations
  - Once deployed, the level of usage is 870% above the predicted levels
  - Usage level is increasing daily
  - The response times are also increasing exponentially
  - Customers are going to the competition because they tired of waiting

Understanding the Business

- What does the business need?
  - What are the problems that need to be solved?
  - Is there enough information to state the problem?
  - What are the specific outcomes that are needed?
- Service Level Agreements
  - What are the quality levels needed?
  - How do we measure outcomes?
  - What is the necessary business outcome?
  - How will it be measured?
- Key Performance Indicators
  - What needs to be measured and why?
  - How do the KPIs relate to SLAs?



Code Management

- Code includes code responsible for presenting the model to the end-user
  - Includes presentation logic, specialized code for the intake of data and the return of model predictions
- Changes to this code must be considered as part of the entire project.
  - Any change to this code has the potential to impact the workload as a whole.
  - “Changing anything changes everything.”
- Maintain the code base using best practices

Code Management

- Best Practices
  - Test Driven Development – write tests before the code
  - Clean code – easy to understand change and modify
  - Specification and documentation – externalize the knowledge required to maintain the code
  - Logging – record significant run time events
  - Code quality – human and automated code reviews
  - Security analysis – STRIDE analysis
  - Error reporting
  - Code management via branches, versions

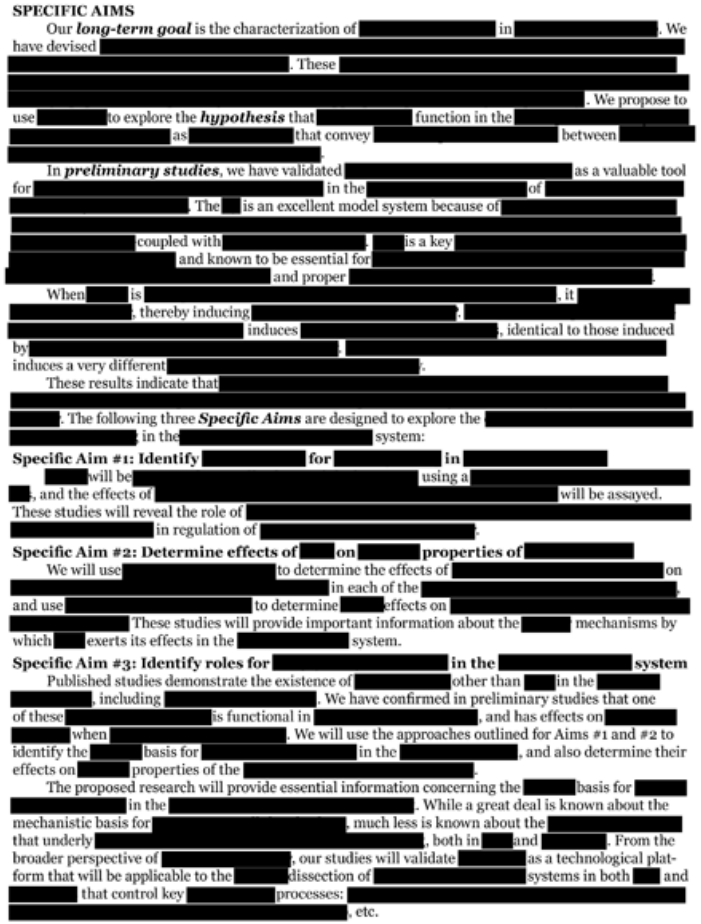
```
import pandas as pd
import numpy as np
import wfdb
import ast

def load_raw_data(df, sampling_rate, path):
    if sampling_rate == 100:
        data = [wfdb.rdsamp(path+f) for f in df.filename_lr]
    else:
        data = [wfdb.rdsamp(path+f) for f in df.filename_hr]
    data = np.array([signal for signal, meta in data])
    return data

path = 'path/to/ptbxl/'
sampling_rate=100

# load and convert annotation data
Y = pd.read_csv(path+'ptbxl_database.csv', index_col='ecg_id')
Y.scp_codes = Y.scp_codes.apply(lambda x: ast.literal_eval(x))

# Load raw signal data
X = load_raw_data(Y, sampling_rate, path)

# Load scp_statements.csv for diagnostic aggregation
agg_df = pd.read_csv(path+'scp_statements.csv', index_col=0)
agg_df = agg_df[agg_df.diagnostic == 1]

def aggregate_diagnostic(y_dic):
    tmp = []
    for key in y_dic.keys():
        if key in agg_df.index:
            tmp.append(agg_df.loc[key].diagnostic_class)
    return list(set(tmp))

# Apply diagnostic superclass
Y['diagnostic_superclass'] = Y.scp_codes.apply(
    aggregate_diagnostic)

# Split data into train and test
test_fold = 10
# Train
X_train = X[np.where(Y.strat_fold != test_fold)]
y_train = Y[(Y.strat_fold != test_fold)].diagnostic_superclass
# Test
X_test = X[np.where(Y.strat_fold == test_fold)]
y_test = Y[Y.strat_fold == test_fold].diagnostic_superclass
```

Data Sources and Exploratory Data Analysis

- Key questions to be answered:
  - What relevant datasets are available?
  - Is this data sufficiently accurate and reliable?
  - How can stakeholders get access to this data?
  - What data properties (known as features) can be made available by combining multiple sources of data?
  - Will this data be available in real time?
  - Is there a need to label some of the data with the “ground truth” that is to be predicted, or does unsupervised learning make sense? If so, how much will this cost in terms of time and resources?
  - What platform should be used?
  - How will data be updated once the model is deployed?
  - Will the use of the model itself reduce the representativeness of the data?
  - How will the KPIs, which were established along with the business objectives, be measured?

Data Constraints

- Key questions to be answered:
  - Can the selected datasets be used for this purpose?
  - What are the terms of use?
  - Is there personally identifiable information (PII) that must be redacted or anonymized?
  - Are there features, such as gender, that legally cannot be used in this business context?
  - Are minority populations sufficiently well represented that the model has equivalent performances on each group?



SPECIFIC AIMS

Our ***long-term goal*** is the characterization of [redacted] in [redacted]. We have devised [redacted]. These [redacted]. We propose to use [redacted] to explore the ***hypothesis*** that [redacted] function in the [redacted] as [redacted] that convey [redacted] between [redacted].

In ***preliminary studies***, we have validated [redacted] as a valuable tool for [redacted] in the [redacted] of [redacted]. The [redacted] is an excellent model system because of [redacted] coupled with [redacted], [redacted] is a key [redacted] and known to be essential for [redacted] and proper [redacted].

When [redacted] is [redacted], it [redacted], thereby inducing [redacted], [redacted] induces [redacted], identical to those induced by [redacted], [redacted] induces a very different [redacted].

These results indicate that [redacted]. The following three ***Specific Aims*** are designed to explore the [redacted], in the [redacted] system:

Specific Aim #1: Identify [redacted] for [redacted] in [redacted]

[redacted] will be [redacted] using a [redacted], and the effects of [redacted] will be assayed. These studies will reveal the role of [redacted] in regulation of [redacted].

Specific Aim #2: Determine effects of [redacted] on [redacted] properties of [redacted]

We will use [redacted] to determine the effects of [redacted] on [redacted] in each of the [redacted], and use [redacted] to determine [redacted] effects on [redacted] These studies will provide important information about the [redacted] mechanisms by which [redacted] exerts its effects in the [redacted] system.

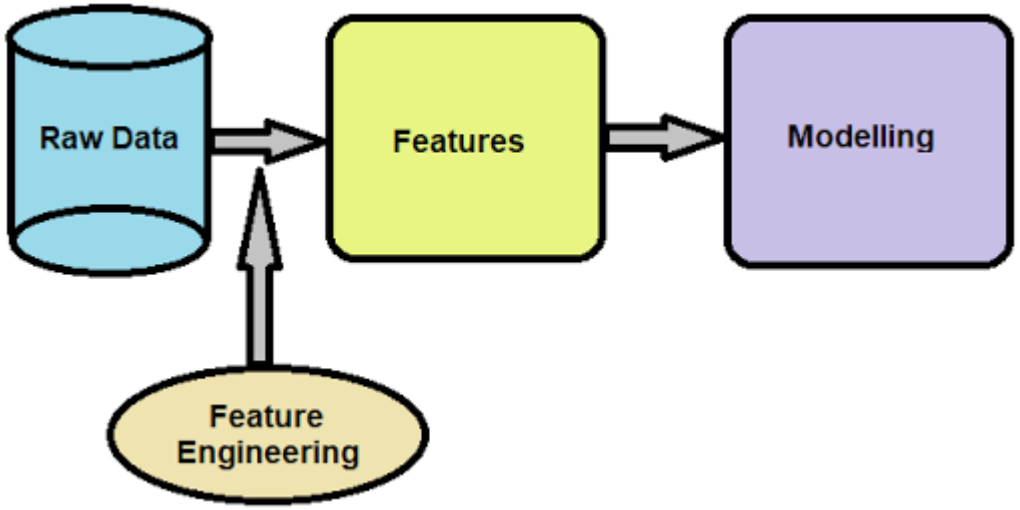
Specific Aim #3: Identify roles for [redacted] in the [redacted] system

Published studies demonstrate the existence of [redacted] other than [redacted] in the [redacted], including [redacted]. We have confirmed in preliminary studies that one of these [redacted] is functional in [redacted], and has effects on [redacted] when [redacted]. We will use the approaches outlined for Aims #1 and #2 to identify the [redacted] basis for [redacted] in the [redacted], and also determine their effects on [redacted] properties of the [redacted].

The proposed research will provide essential information concerning the [redacted] basis for [redacted] in the [redacted]. While a great deal is known about the mechanistic basis for [redacted], much less is known about the [redacted] that underly [redacted], both in [redacted] and [redacted]. From the broader perspective of [redacted], our studies will validate [redacted] as a technological platform that will be applicable to the [redacted] dissection of [redacted] systems in both [redacted] and [redacted] that control key [redacted] processes: [redacted], etc.

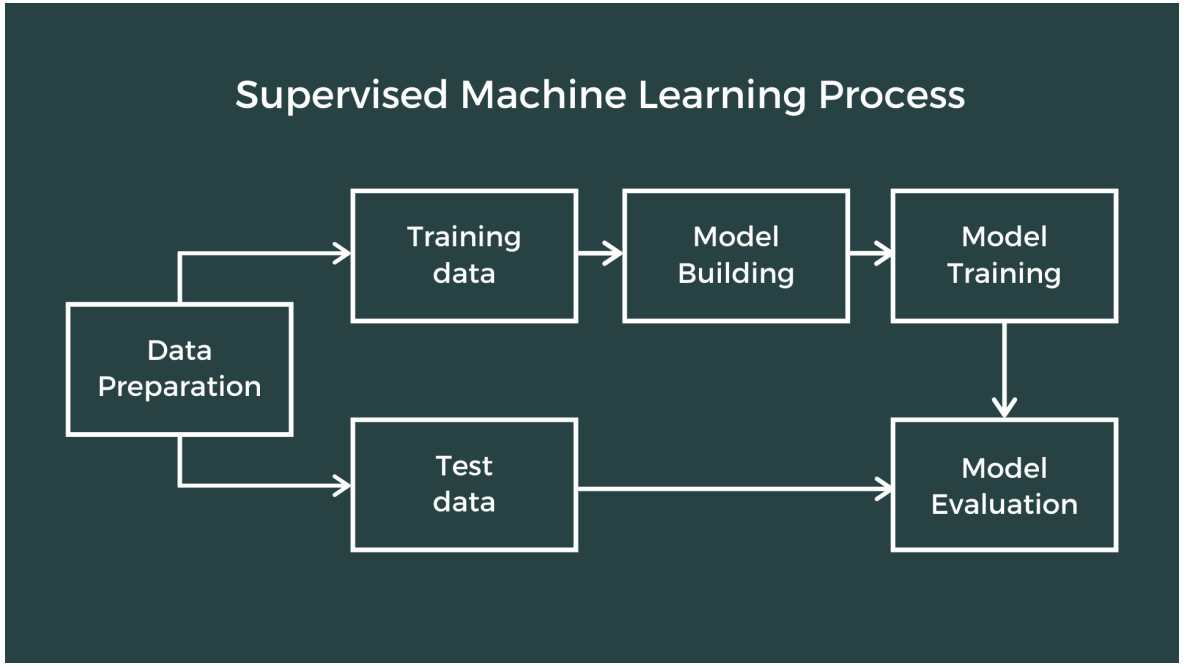
Feature Engineering and Selection

- Critical part of model development
- Poor feature selection impacts MLOps in a variety of ways
  - The model can become more and more expensive to compute.
  - More features require more inputs and more maintenance down the line.
  - More features mean a loss of some stability.
  - The sheer number of features can raise privacy concerns.



Training and Evaluation

- Standard methodology in ML
- For MLOps
  - Is the training data realistic?
- We train a model on fixed data
  - Tune results by algorithm choice
  - Tune results by feature selection
  - Tune results with hyperparameters
- Overfitting is real concern
- Setting evaluation criteria is important
  - Moderate success on wide ranges of training data versus high success on narrow sets of training data



Reproducibility

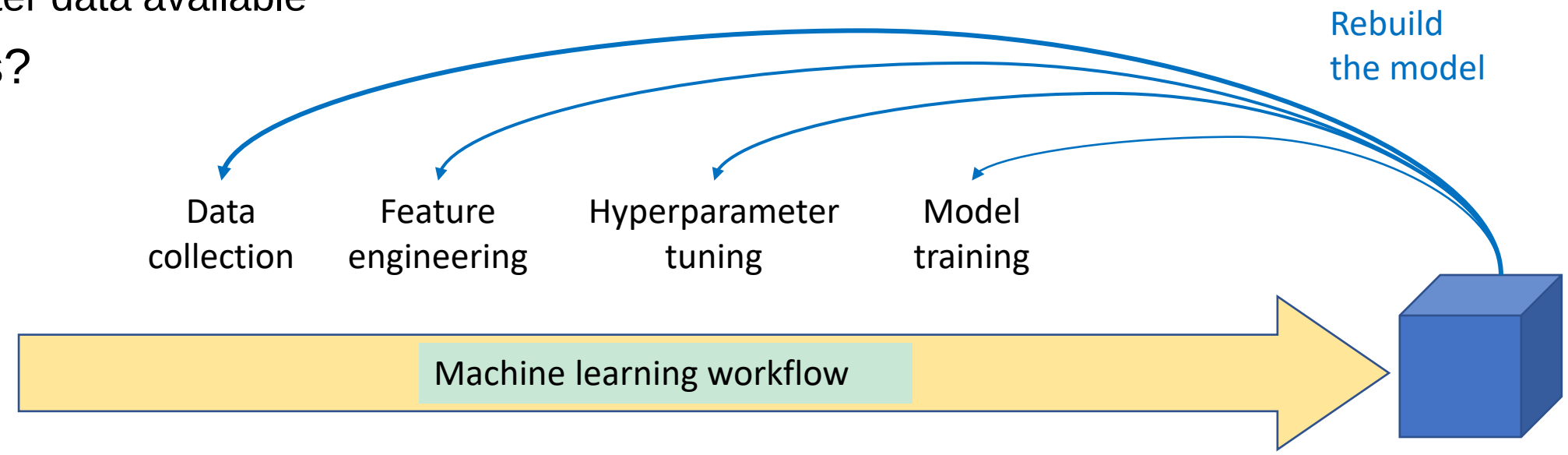
- Supported by good CI practices
- What was created in dev can be reproduced in prod
- Ability to roll back to previous versions
- Requires version control of ALL assets used in developing the model

Responsible AI

- Results need to be explainable
- Mitigate uncertainty and help prevent unintended consequences
- Uses:
  - Partial dependence plots, which look at the marginal impact of features on the predicted outcome
  - Subpopulation analyses, which look at how the model treats specific subpopulations and that are the basis of many fairness analyses
  - Individual model predictions, such as Shapley values, which explain how the value of each feature contributes to a specific prediction
  - What-if analysis, which helps the ML model user to understand the sensitivity of the prediction to its inputs

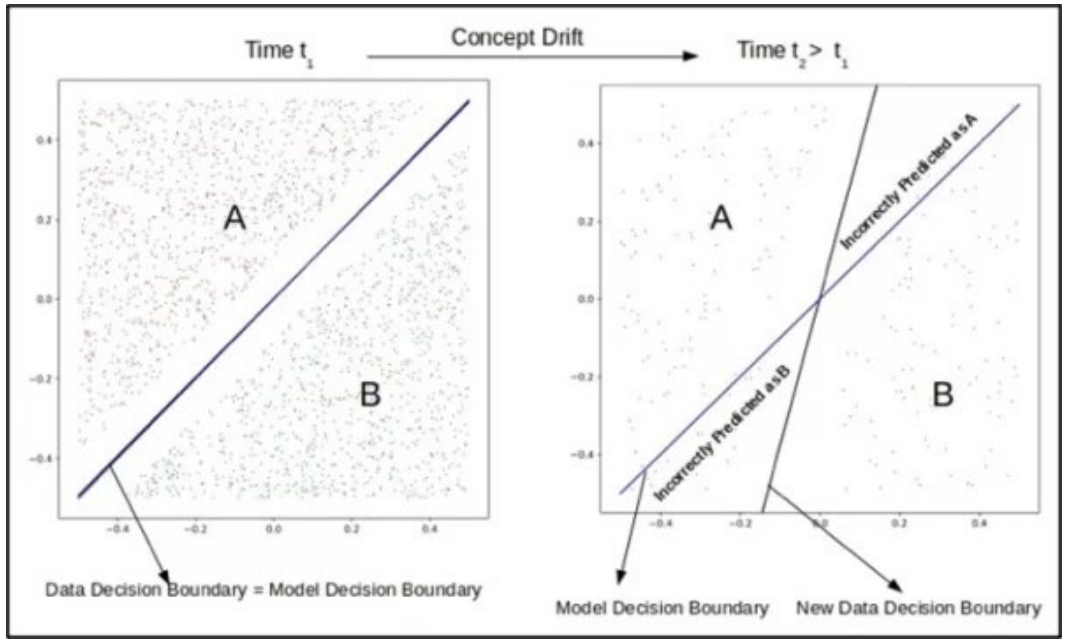
Model Rebuild

- Reasons for a model rebuild
  - Changing business requirements
  - Data drift
  - Concept drift
  - Better data available
- Others?



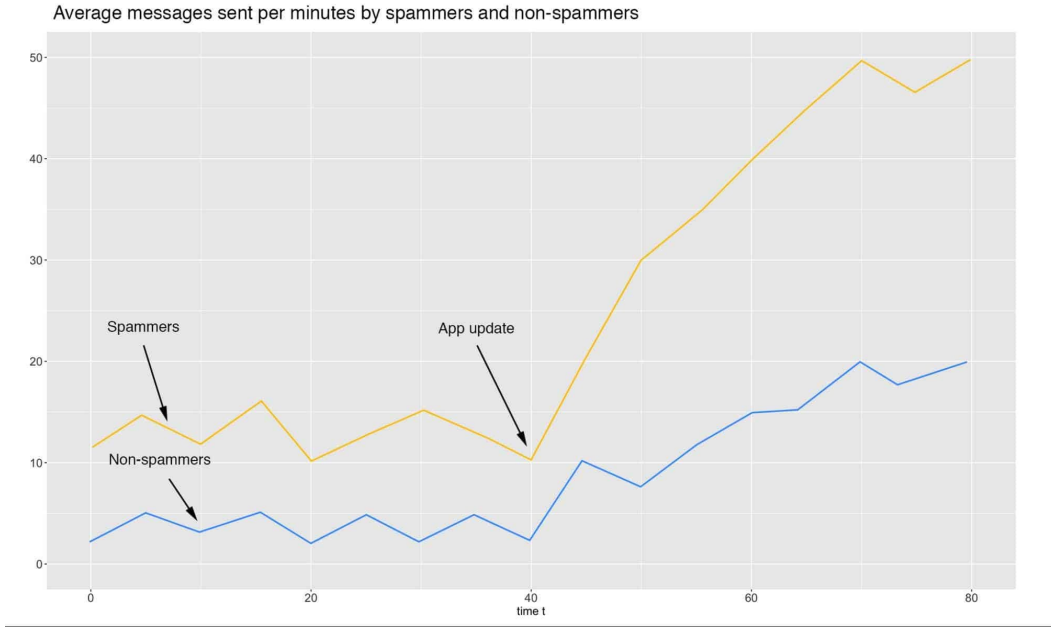
Concept Drift

- ML models are trained assuming a certain conceptual relationship between inputs and predictions
  - When that relationship changes in the real world, we have concept drift
  - The model no longer describes the actual relationships
  - For example, floor area is no longer a factor in home prices.
  - Indicates a change in the decision boundary
- Concept drift can be managed only by retraining the model



Data Drift

- Key ML assumption
  - Future is predictable from the past
- Data drift is when
  - The distribution of the input data has changed
- Temporal shifts
  - The population changes over time – changing demographics for example
  - Features in the model no longer correspond to features in the data
- Spatial shifts
  - The model is deployed in a different population than it was trained for
  - US models deployed in China



ML: Fundamental Parts

Business

Data

Build

- Build
- Train
- Evaluate

Deploy

- Deploy
- Manage

Monitor

- Predict
- Monitor

ML: Business

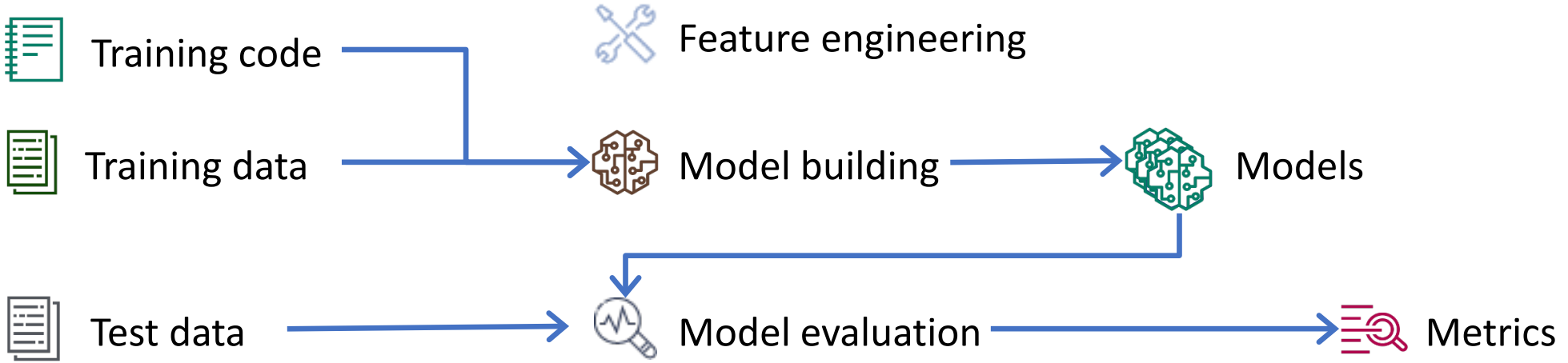
- Business problem
- ML framing of the problem
- Discovery
- Key Performance Indicators (KPIs)

ML: Data

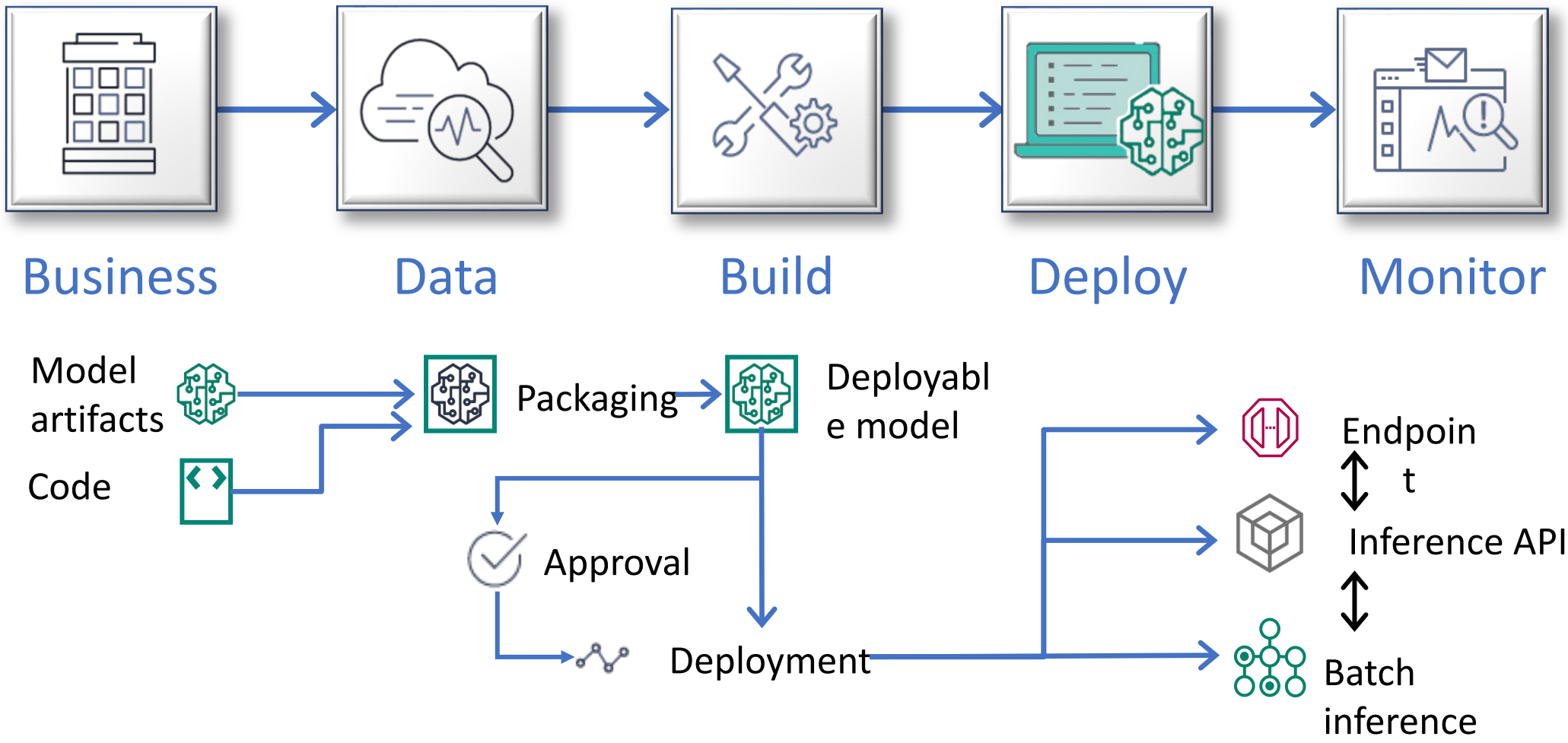
- Collection
- Preparation
- Analysis
- Data strategy

ML: Build

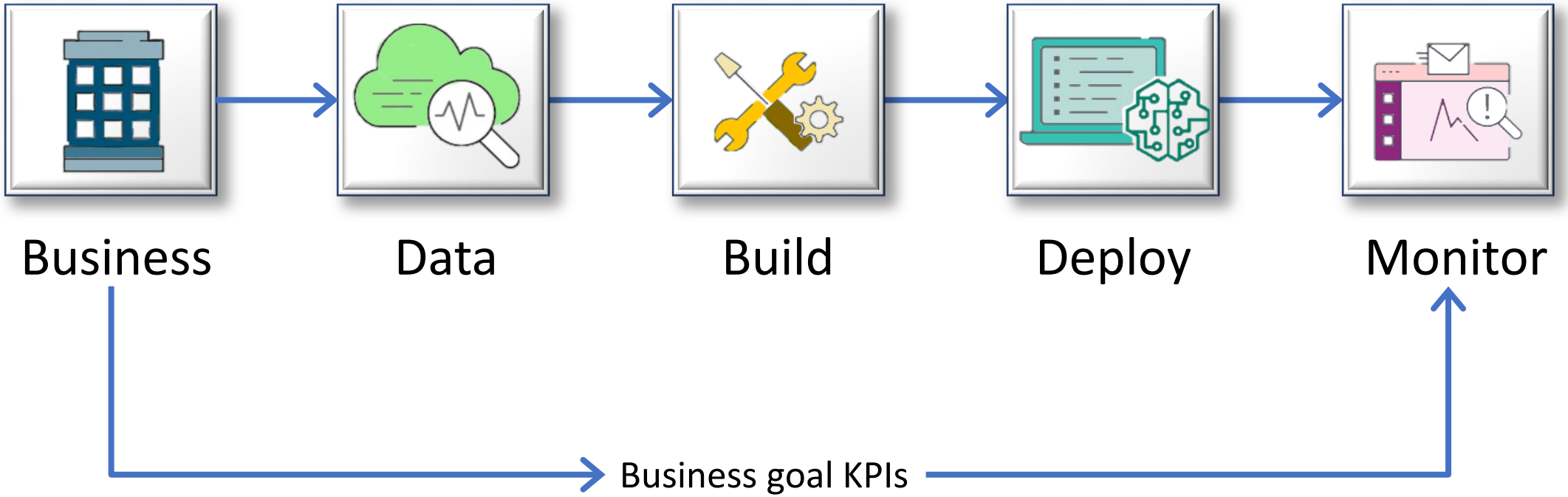

Business
Data
Build
Deploy
Monitor


Training code
Feature engineering
Training data
Model building
Models
Test data
Model evaluation
Metrics

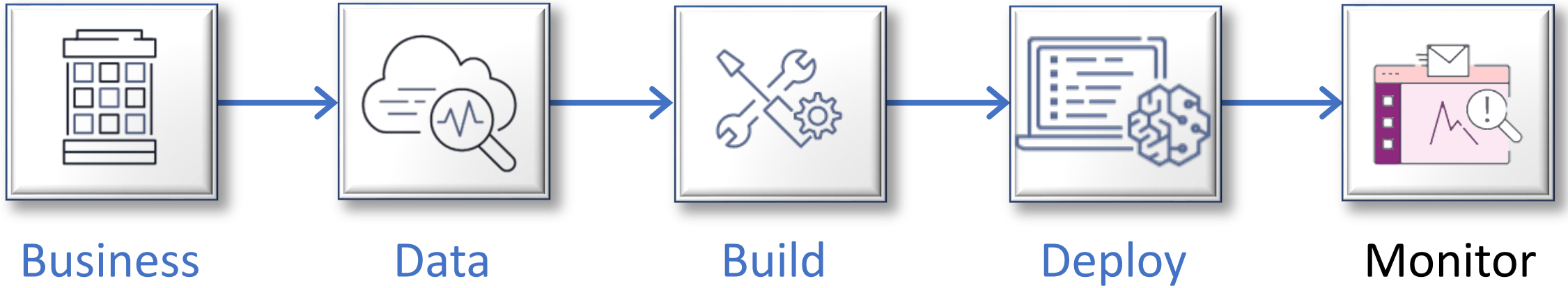
ML: Deploy

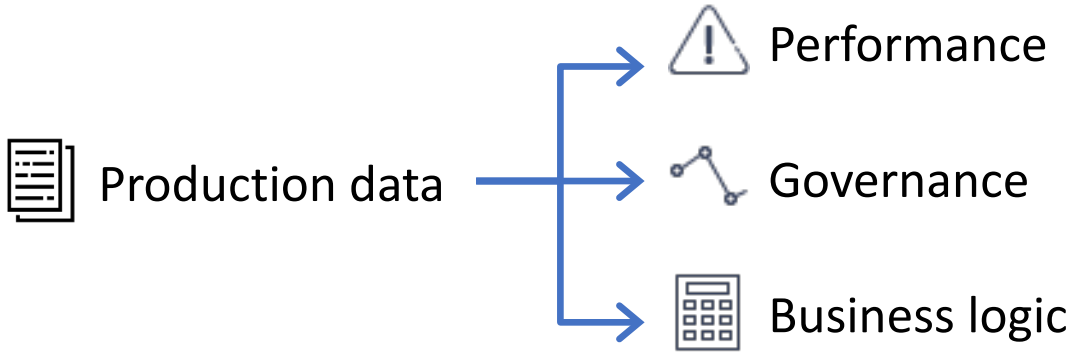

Business
Data
Build
Deploy
Monitor
Model artifacts
Code
Packaging
Deployable model
Approval
Deployment
Endpoint
Inference API
Batch inference

ML: Business Goal



ML: Monitoring





Monitoring

- Performance monitoring
  - Instrumentation at the endpoint determines whether app is meeting KPIs
    - E.g. response time or throughput of the entire run path
    - Might include measures of the effectiveness of the model itself, measuring metrics such as accuracy, f1 score, or root mean squared error (RMSE).
- Governance monitoring
  - Data governance – Processes designed to ensure the appropriate use and access management of data
  - Process governance – Implementation of policies and processes to ensure governance issues are identified and addressed at appropriate times throughout the ML workflow.
- Business logic monitoring
  - Verifies that the solution continues to meet the original business goals.
  - Explainability - For example, the source data might be seasonal, but the predictions are supposed to be de-seasonalized.

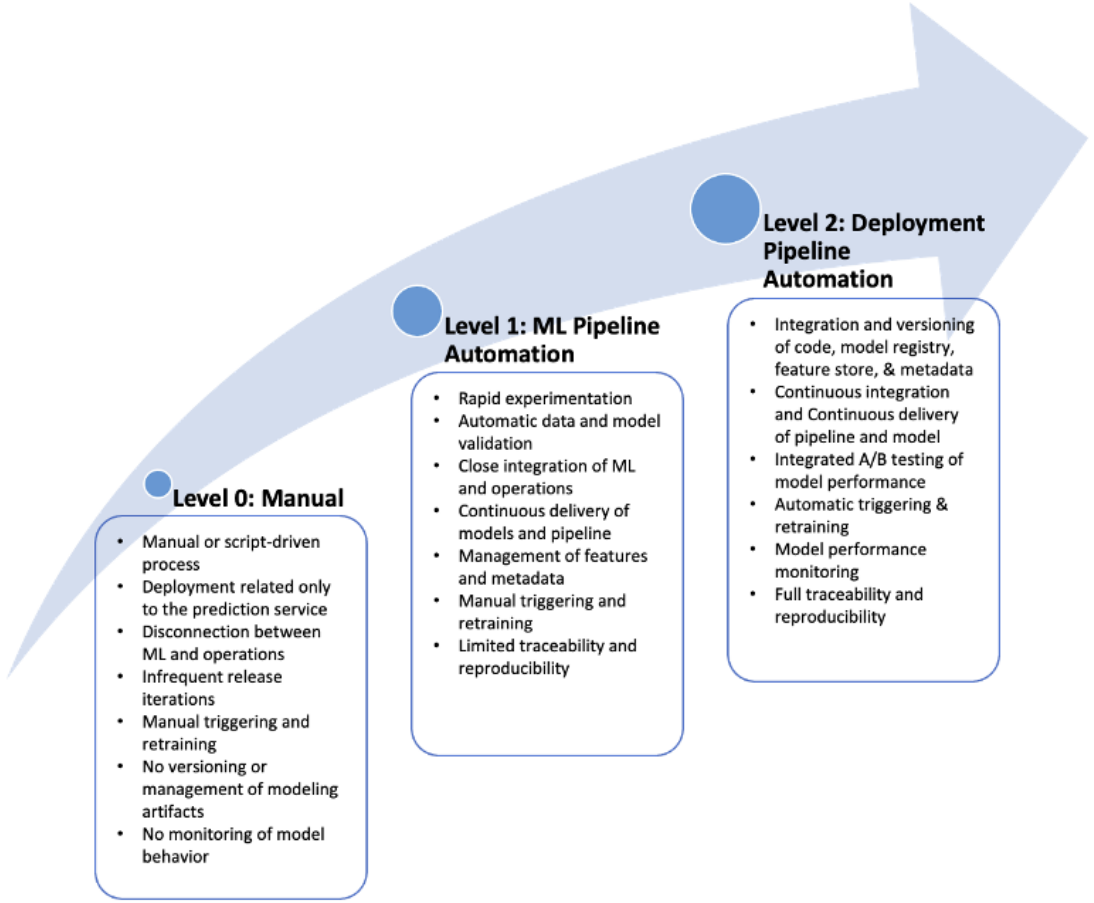
Monitoring

- Performance monitoring
  - Instrumentation at the endpoint determines whether app is meeting KPIs
    - E.g. response time or throughput of the entire run path
    - Might include measures of the effectiveness of the model itself, measuring metrics such as accuracy, f1 score, or root mean squared error (RMSE).
- Governance monitoring
  - Data governance – Processes designed to ensure the appropriate use and access management of data
  - Process governance – Implementation of policies and processes to ensure governance issues are identified and addressed at appropriate times throughout the ML workflow.
- Business logic monitoring
  - Verifies that the solution continues to meet the original business goals.

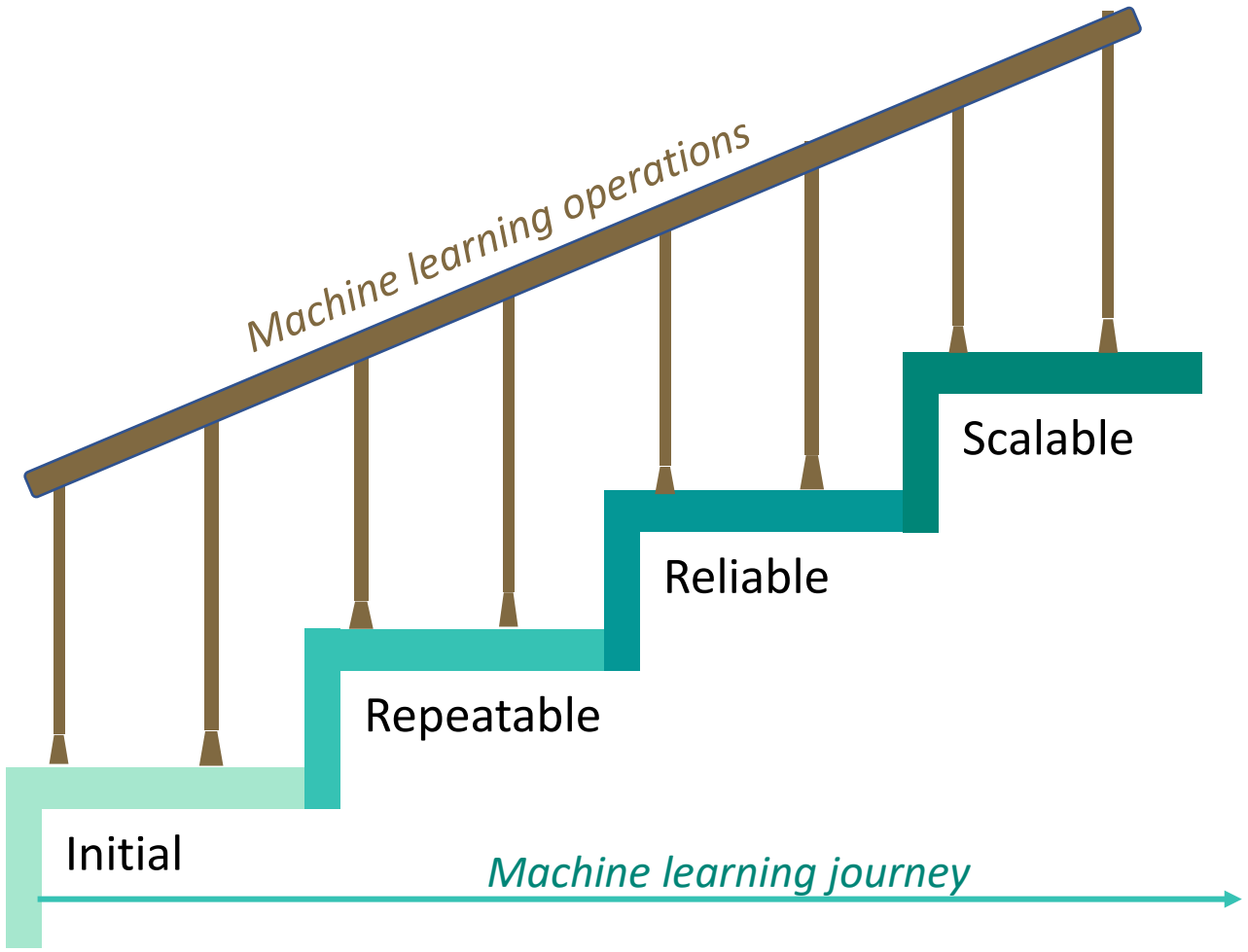
MLOps Maturity Levels

Maturity Levels

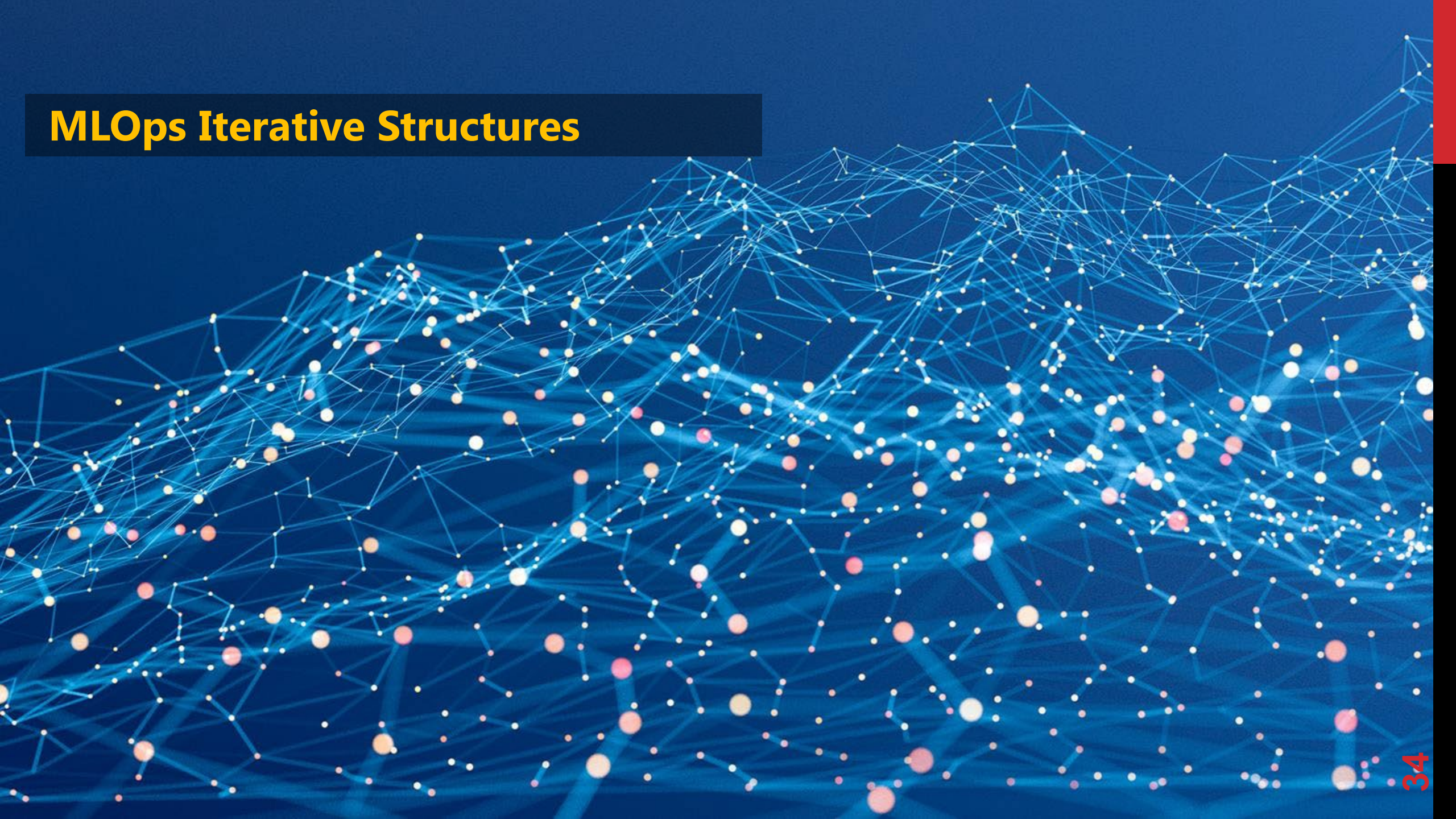
- Various attempts to quantify the organization's level of expertise in using MLOps
  - https://www.eqengineered.com/insights/mlops-maturity-levels
  - https://docs.microsoft.com/en-us/azure/architecture/example-scenario/mlops/mlops-maturity-model



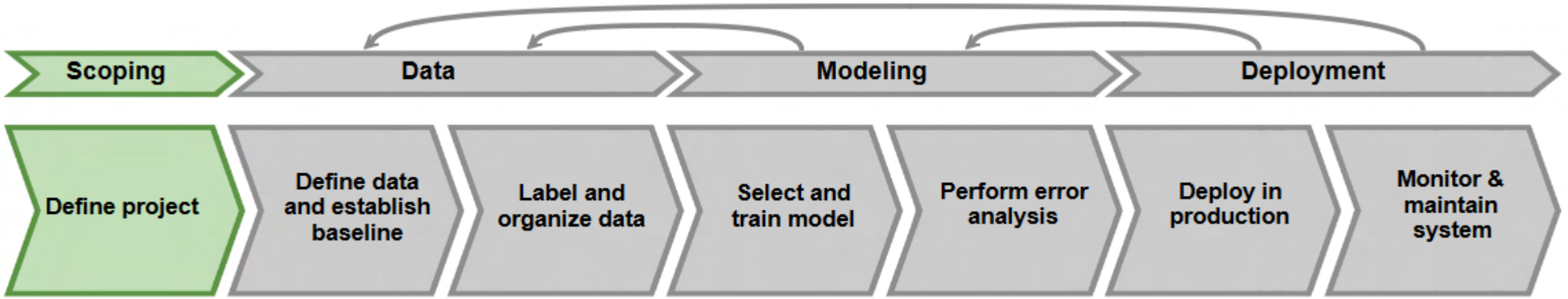
Process Maturity



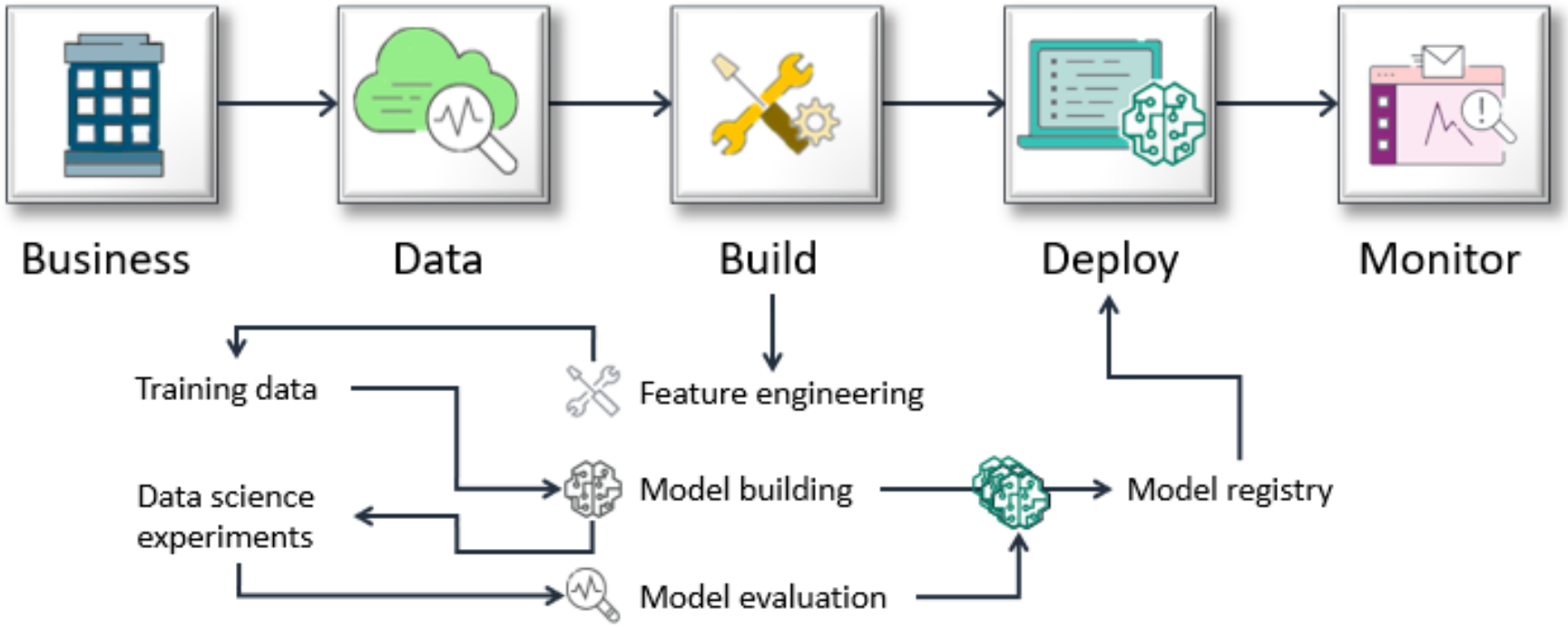
MLOps Iterative Structures



Model Lifecycle



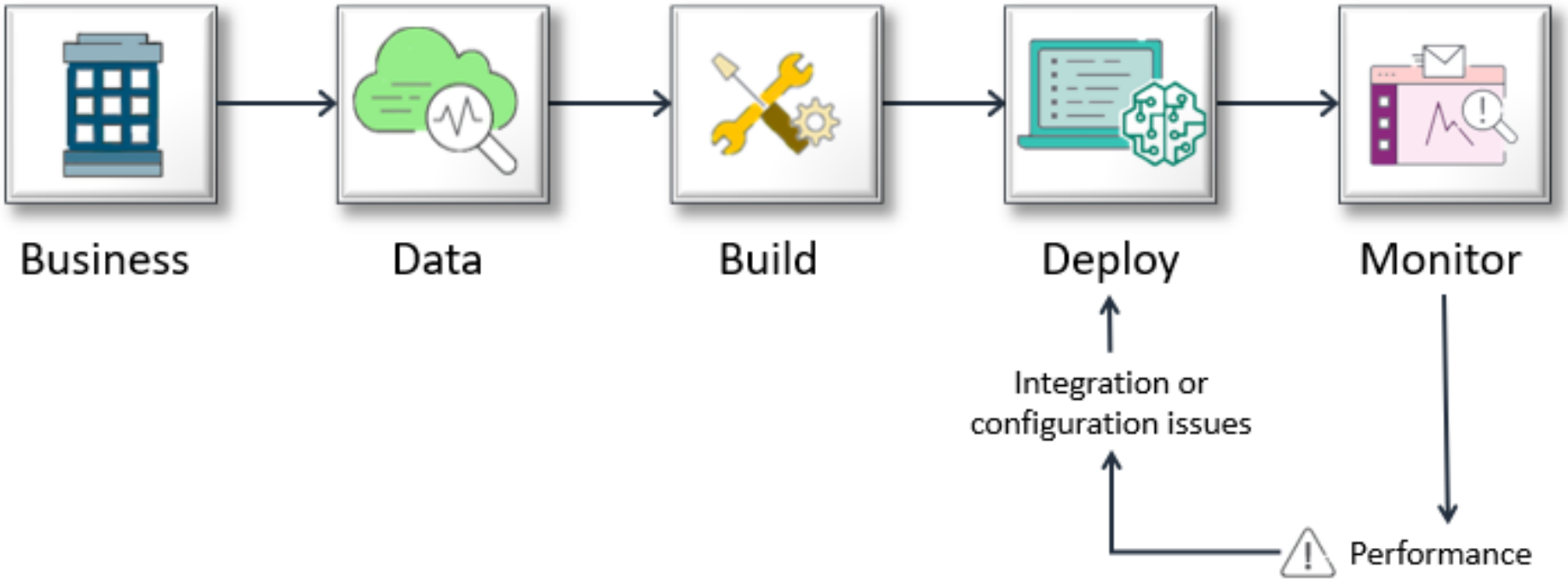
Build and Deploy Loops



A data scientist selects features and processes for the training data, to surface those features and make them accessible for model training. That data is one input into the model building activity.

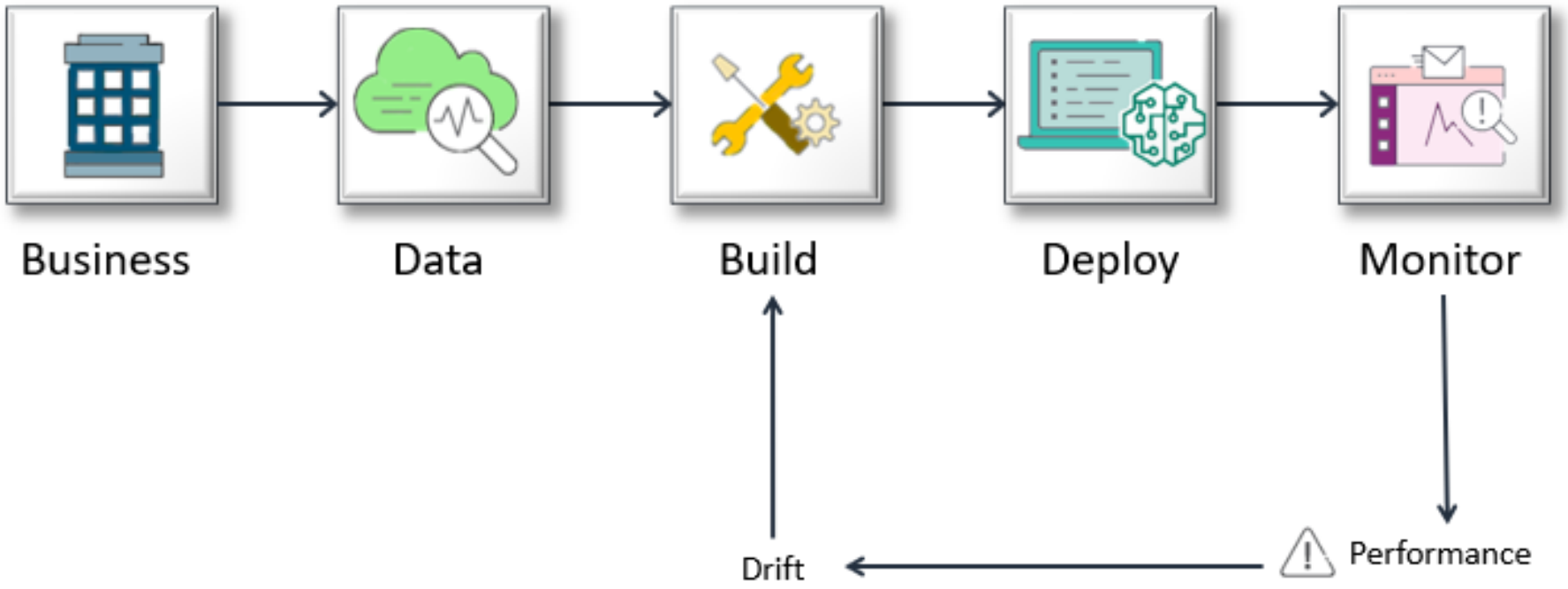
The data scientist iteratively experiments, building models and testing them. One or more candidate models might have predictive capacity and other required qualities. These models are evaluated. Candidate models are placed into the model registry. Models are selected for deployment. These models become an input into the packaging activity.

Monitor Feedback Loop



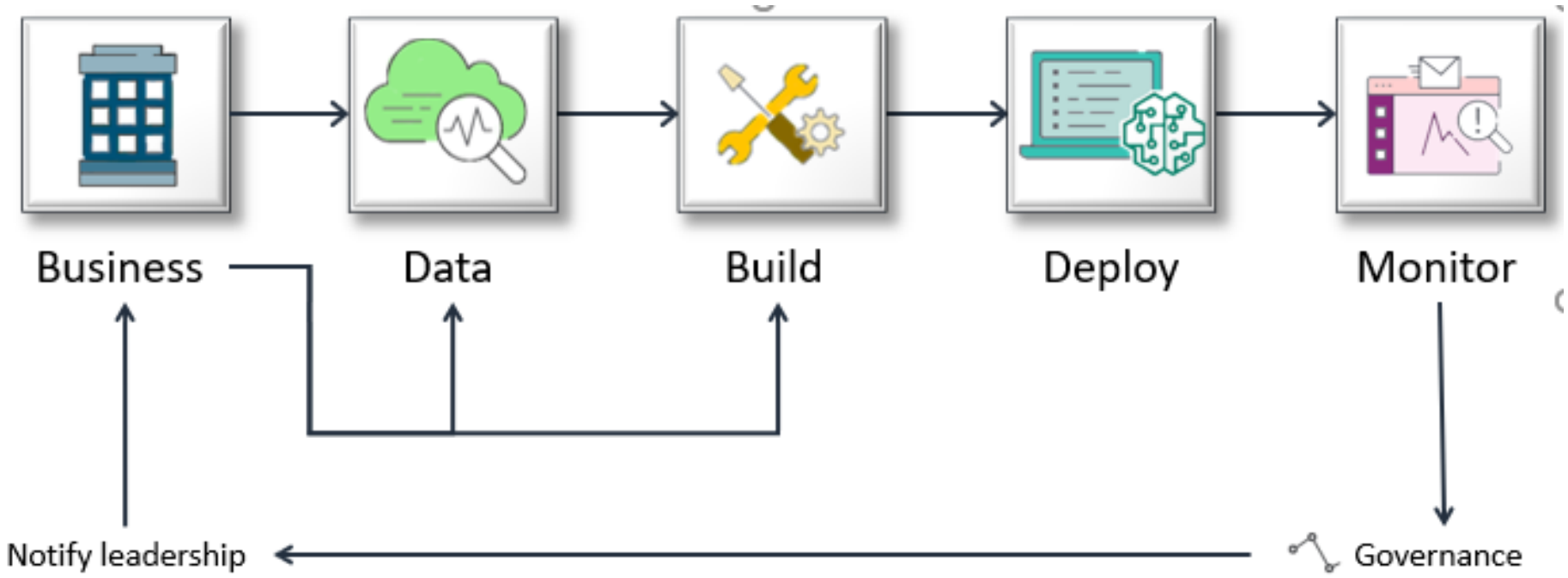
Information from monitoring the performance of the inference is used to determine whether the values indicate that there are issues, possibly with application integration or endpoint configuration. Logs will sometimes indicate issues, even if they were not reported by consumers of the inference.

Inference Drift



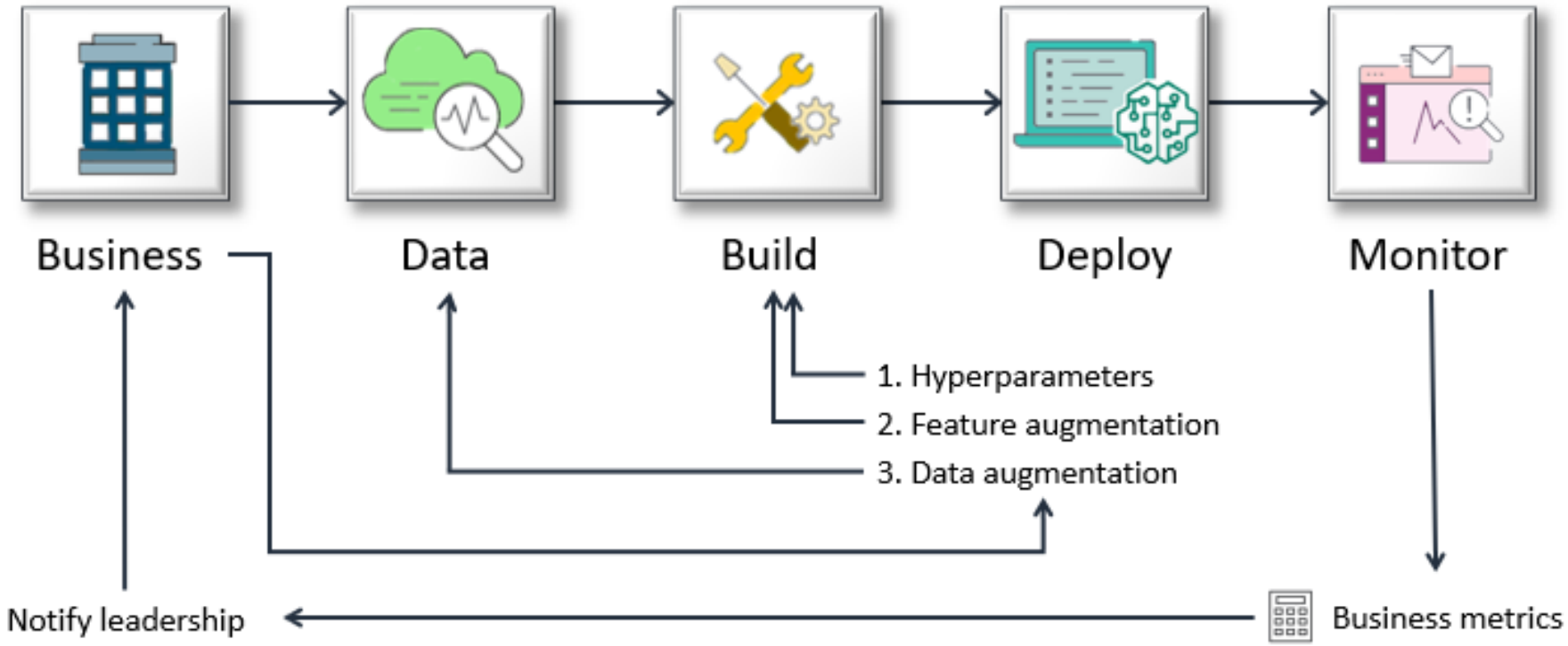
Information reported by the consumer of the inference is used to determine whether the inference indicates possible inference drift. In this case, the next step is to attempt to fix the inference using feature engineering or additional model tuning. If that does not work, the next step is to address the issue by doing something to improve or augment the training data. If that does not work, the final step is to report the problem back to business leadership to determine how to proceed.

Governance Feedback Loop



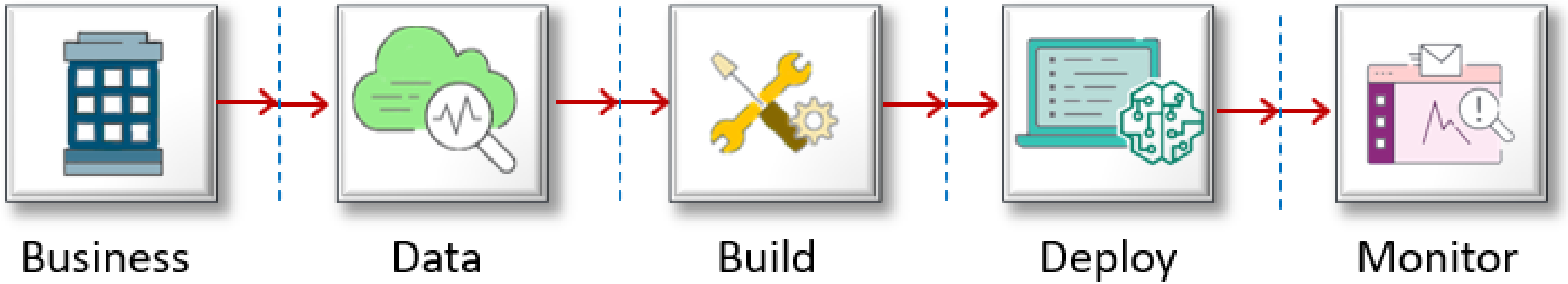
Governance monitoring detects if the service is at risk of going out of compliance, and then it notifies business leadership. This includes explainability reporting, post-deployment monitoring, and requirements monitoring. Leadership can then take action to mitigate the issue, which could involve data augmentation, feature augmentation, or model rebuild. Other business-level actions might also be taken.

Business Goals Feedback Loop

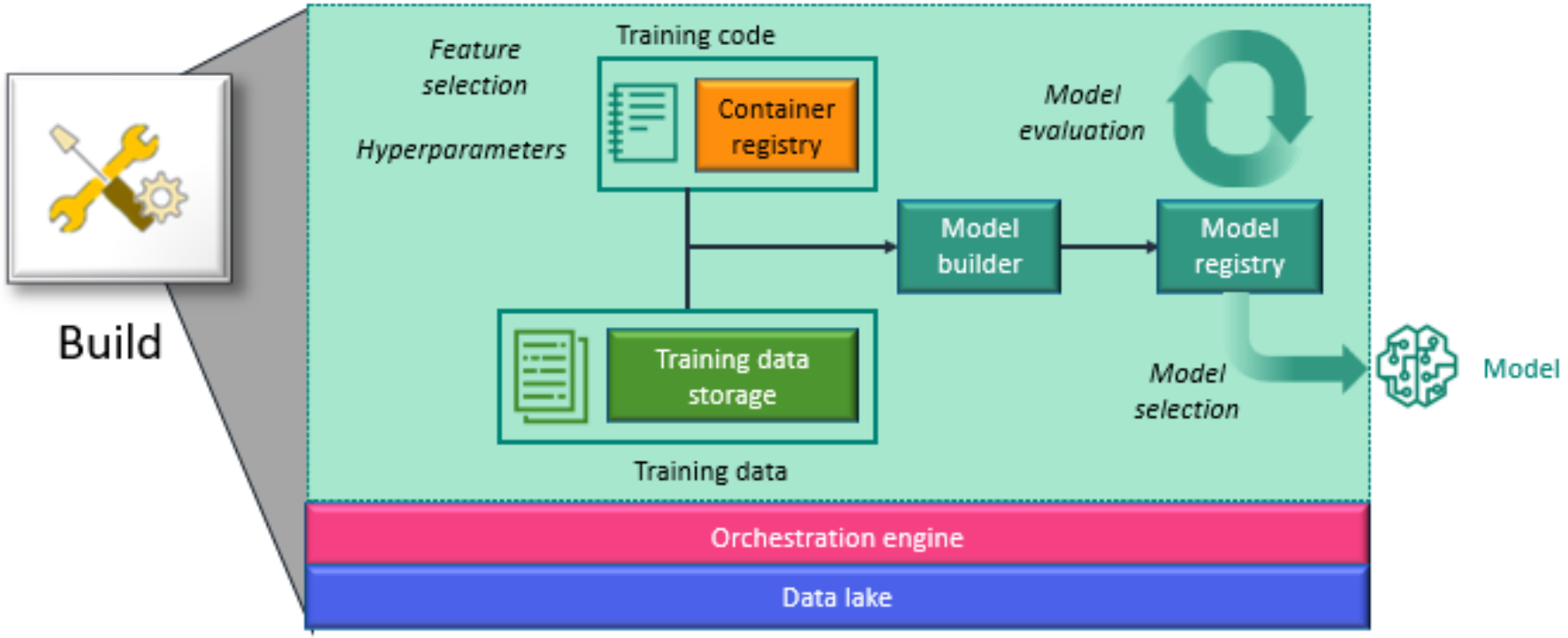


Occurs when the model is no longer effectively meeting the business goals outlined for the application. Depending on what is determined to have caused the model to not meet the business measurement, it might mean having to return to different phases of the development cycle. For example, retraining of the model to see if that brings it back in line with the business metric. For example, there might be seasonality in the data for sales forecasts, or perhaps new entertainment has been added or removed from a favorite streaming service. Individual business rules would also determine what level of notifications, back to the interested business leadership, would be required.

Quality Gates in ML Workflow



Inside a Build, Train, Evaluate Workflow



Inside a Build, Train, Evaluate Workflow

- The orchestration engine coordinates model training and evaluation
  - During the model development phase, a data scientist will train and tune models through data collection processing and analysis, model training, tuning and evaluation, and (eventually) creating a deployable artifact.
  - The orchestration engine might include automation of model training, tuning through a model builder environment (such as SageMaker), and model evaluation that might choose one model or multiple models to deploy.
  - Orchestration might be limitied to the deployment of trained model artifacts.
    - ML best practices typically include integration with a model registry, which is a service that manages model artifacts.
    - It also tracks which models are deployed in production, and it tracks the associated model metadata.
  - The level of orchestration might also dictate the level of data store or data lake integration.

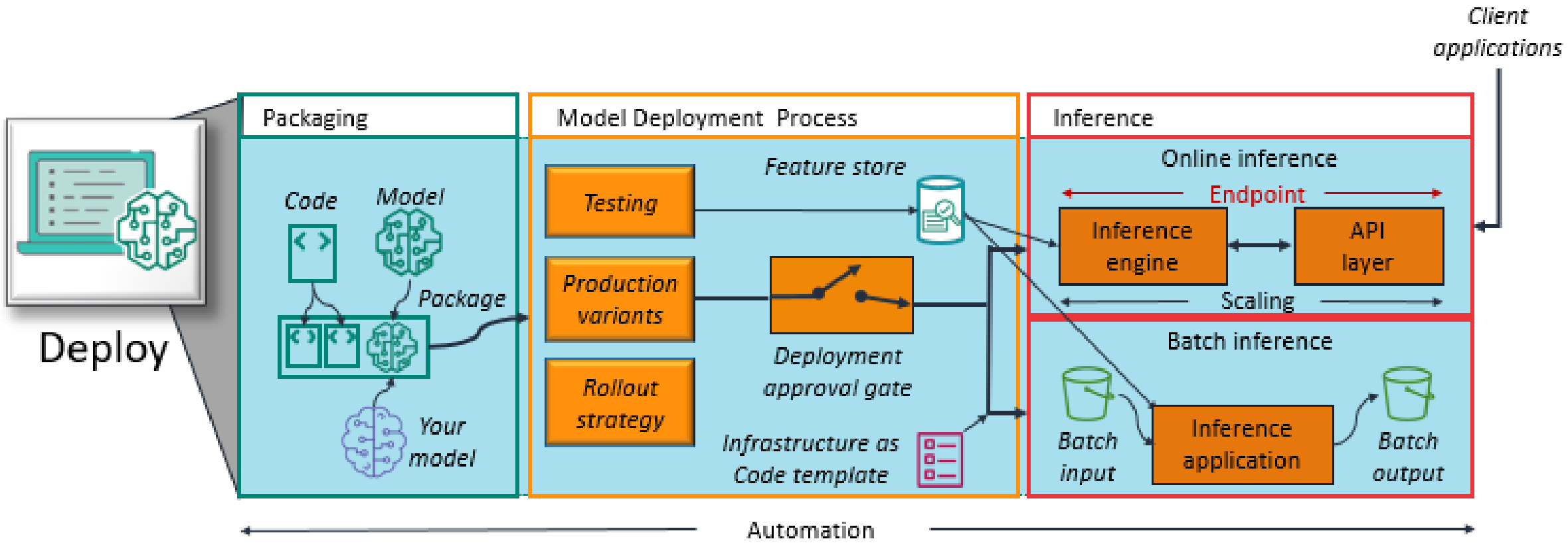
Inside a Build, Train, Evaluate Workflow

- The training code is input to a model builder, such as SageMaker, along with training data.
  - The training data could be staged within a data lake.
  - In AWS, a typical location for staging training data is in Amazon S3.
  - The model builder implements ML algorithms and trains the model.
  - The results are typically tracked as an experiment with a pointer to the artifact, such as a reference to the model in S3.
  - Alternatively, metadata for models to be deployed, such as the pointer to the model, is stored in a model registry.

Inside a Build, Train, Evaluate Workflow

- Typically, a data scientist evaluates the test model for predictive accuracy, bias, and other qualities.
  - Changes are made to training code and training data, and additional test models are generated.
  - The data scientist engages in an experimentation process, using the scientific method in which experiments are defined and trials are performed.
  - This experimentation process is iterative until one or more models are generated that meets the criteria for production.
  - A few models are selected, from the candidate models, for deployment to production.
- A central theme in MLOps is reproducibility through automation.

Inside a Deploy and Manage Workflow



Inside a Deploy and Manage Workflow

- The output from the packaging phase could be the trained model artifact - model deployment process simply is responsible for making the model available to web server. (Continuous Delivery)
  - Emerging best practices for MLOps include the use of model registries to track metadata for the trained models.
  - The deployment process interfaces with the model registry to determine downstream decisions about the deployment targets.
- The model deployment process might also have manual or automated approval gates, especially for production deployments.
  - In some cases, the model deployment might actually include the deployment of multiple models to be accessible as a single endpoint.
  - There are different processes for accomplishing this.
  - The goal of the deployment step is to utilize the different inputs (such a testing definitions, production variants, rollout strategies, and so on) to deploy the trained model or models in such a manner that they can be integrated into client or batch applications.

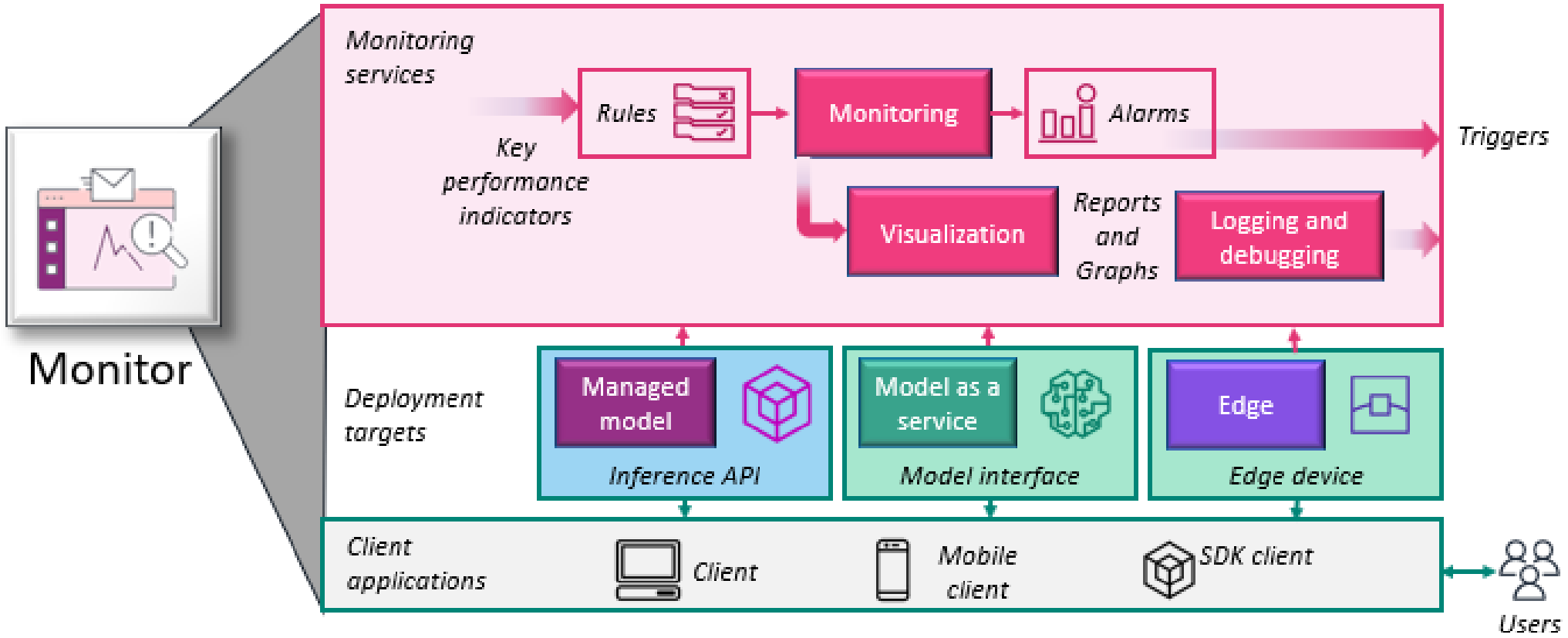
Inside a Deploy and Manage Workflow

- Automation
  - A common method in AWS is to use AWS Step Functions to coordinate the process.
- Staging
  - The selected model metadata is accessed from the model registry.
  - Various kinds of code might be included, depending on deployment requirements.
  - Helper code, used to complete deployment, is an example.
  - Inference code, used within an API, is another example.
  - This code could be staged in several locations.
  - S3 is a common location for staging code in AWS.
  - The deployment process packages the model and code. It might perform tests to verify that the package is ready for deployment.

Inside a Deploy and Manage Workflow

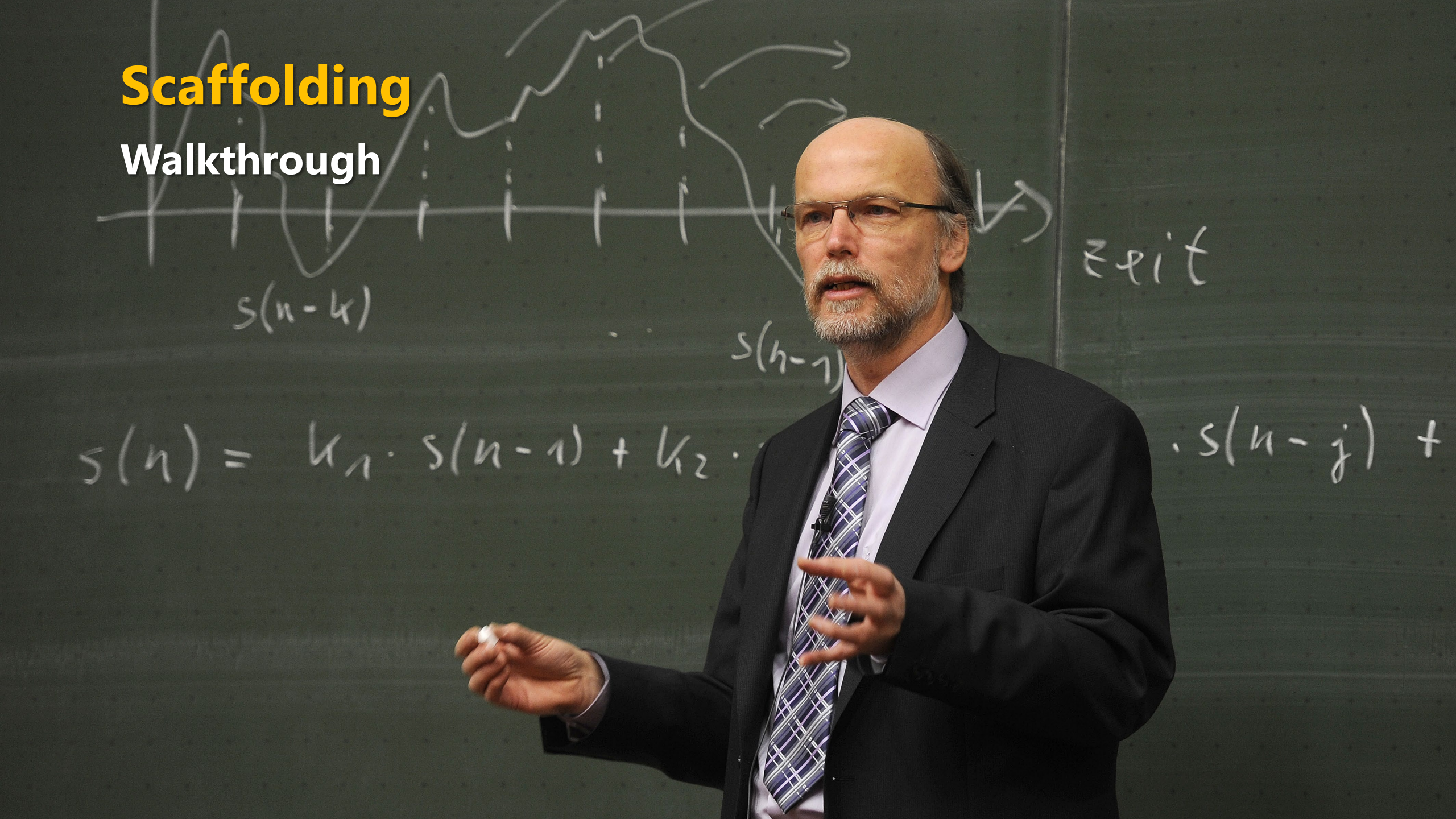
- Approval
  - After the package is created and verified, the next step is an approval gate where leaders in DevOps, business leaders, or a data scientist determine that it is the right time to deploy the package.
- Deployment
  - After approval, the package is deployed to deployment targets. Client applications can then use inferences from the newly deployed model and code.

Inside a Predict and Monitor Workflow



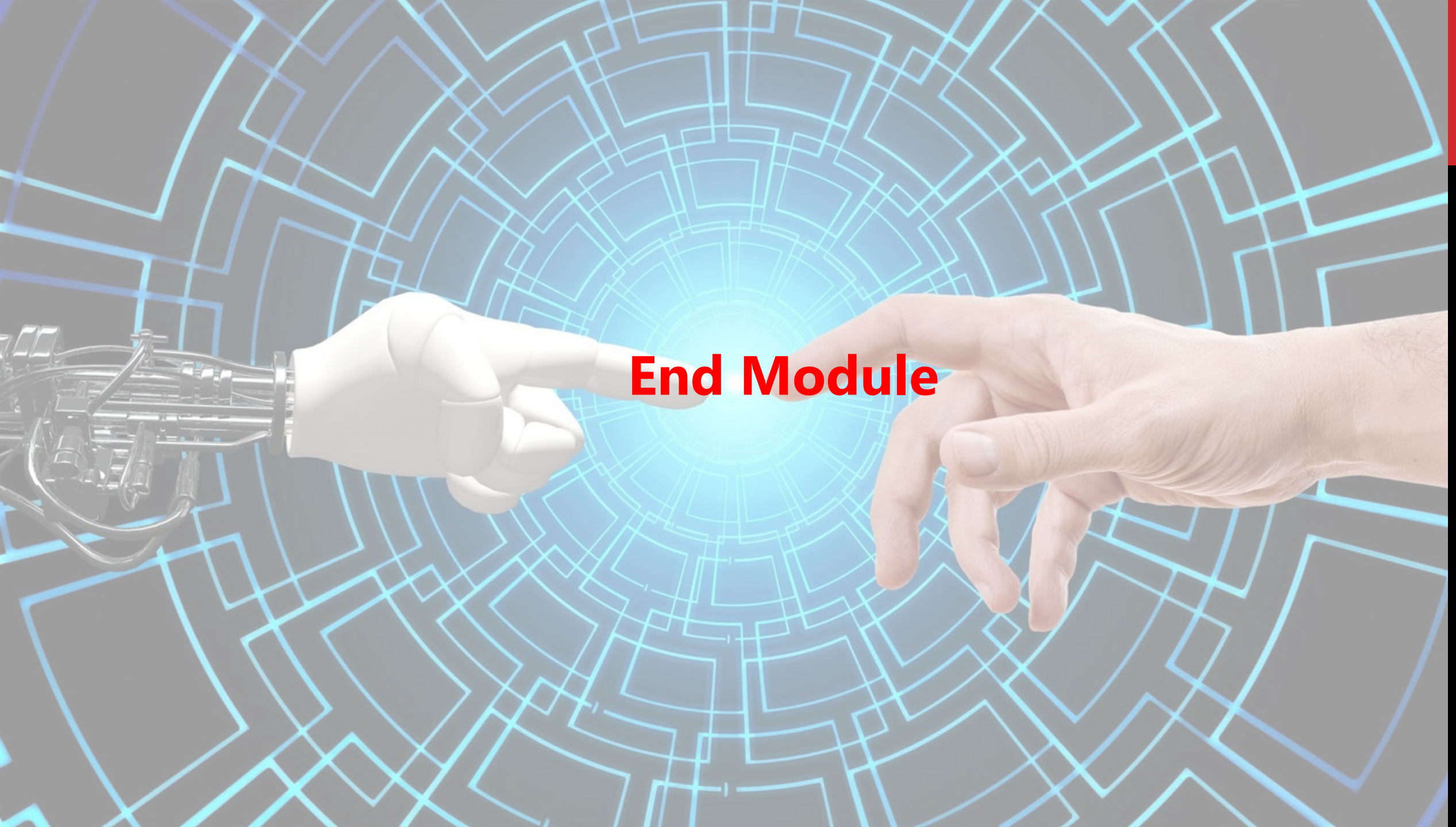
Inside a Predict and Monitor Workflow

- A managed model can be run on the cloud or on premises.
  - It might use orchestration engines or developed in house
  - Model as a service - prebuilt models that are offered with a documented interface that could be running on premises, in a third-party location, or in AWS.
  - In edge devices, the model is deployed to an end device for local inference.
- Operational metrics are reported from deployment targets
  - KPIs are formalized into monitoring rules. The rules are applied to the incoming metrics to determine when to issue alarms.
  - Alarms are triggers to automated processes or manual procedures. The manual procedures might be contained in a runbook.
  - Monitoring also provides metrics data for visualization and business intelligence in the form of reports and graphs.
  - Logging and debugging information is also collected, which can be useful when an alarm invokes a deeper analysis of operations.


Scaffolding
Walkthrough

Scaffolding

Lab 3


End Module