MLOps

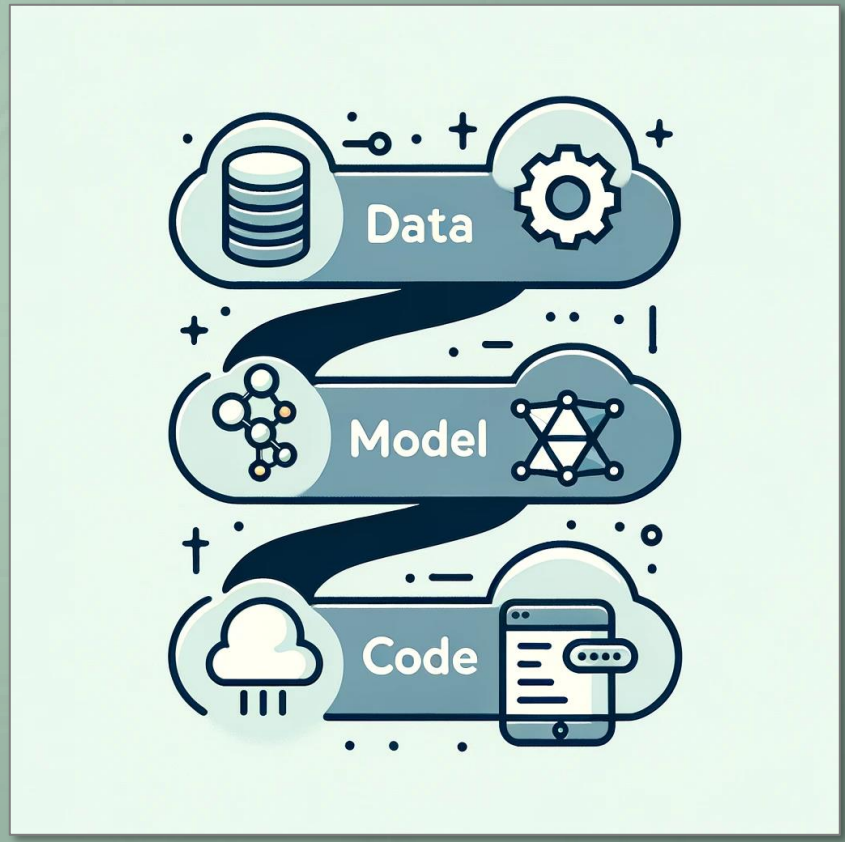
Advanced Software-Engineering

Dr. Harald Stein
Prof. Dr. Selcan Ipek-Ugay

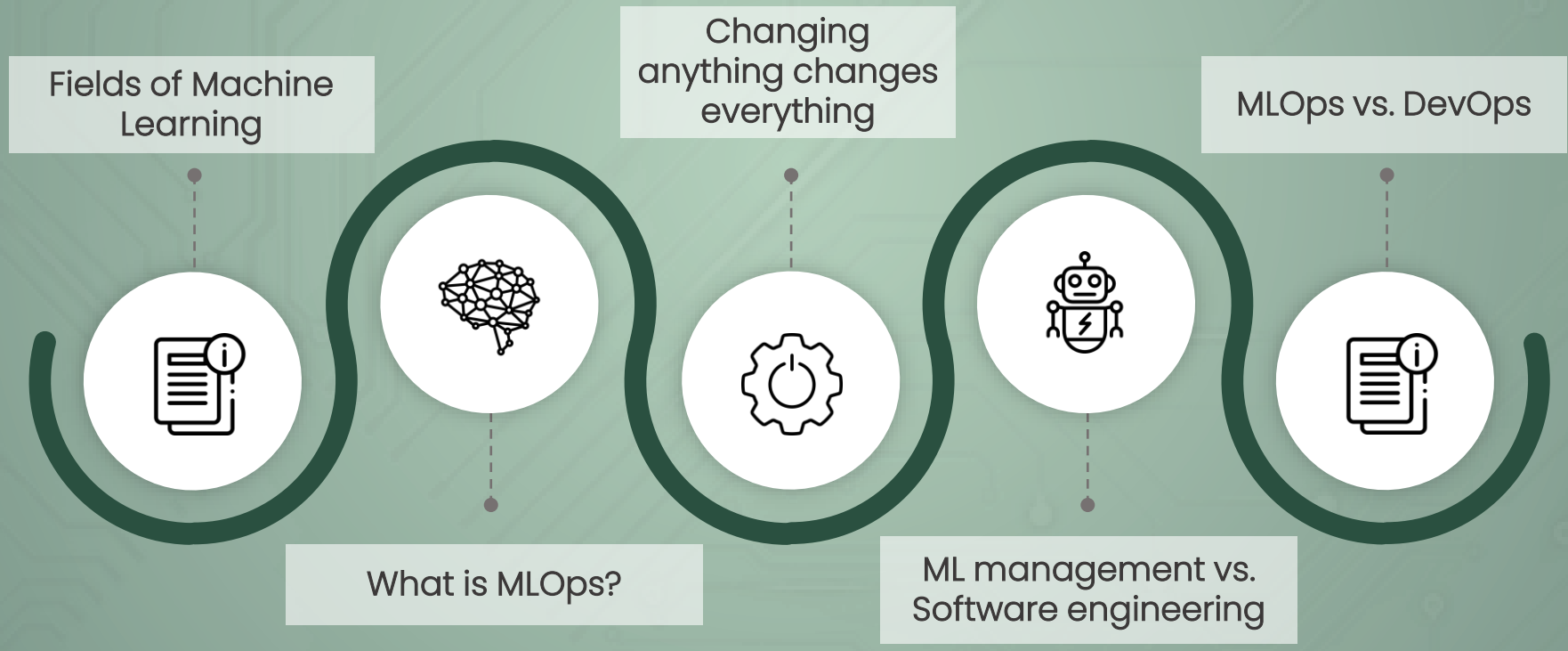
WS 2024/25

Agenda

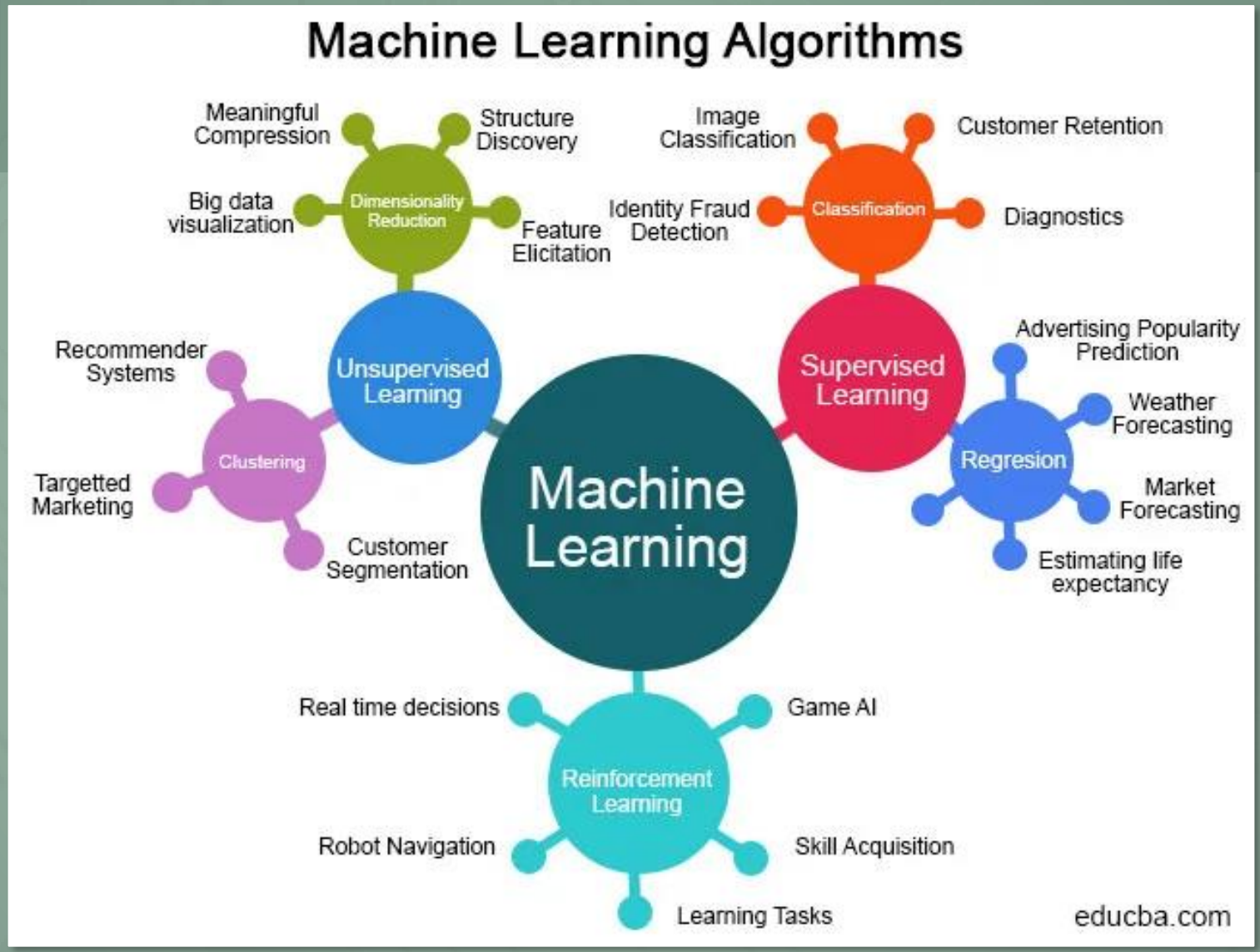
- What is ML Ops?
- Business Problems requiring ML
- Overview of Machine Learning engineering
- ML Ops Principles
- Quality Assurance & Governance
- Technology & Examples



What is MLOps?

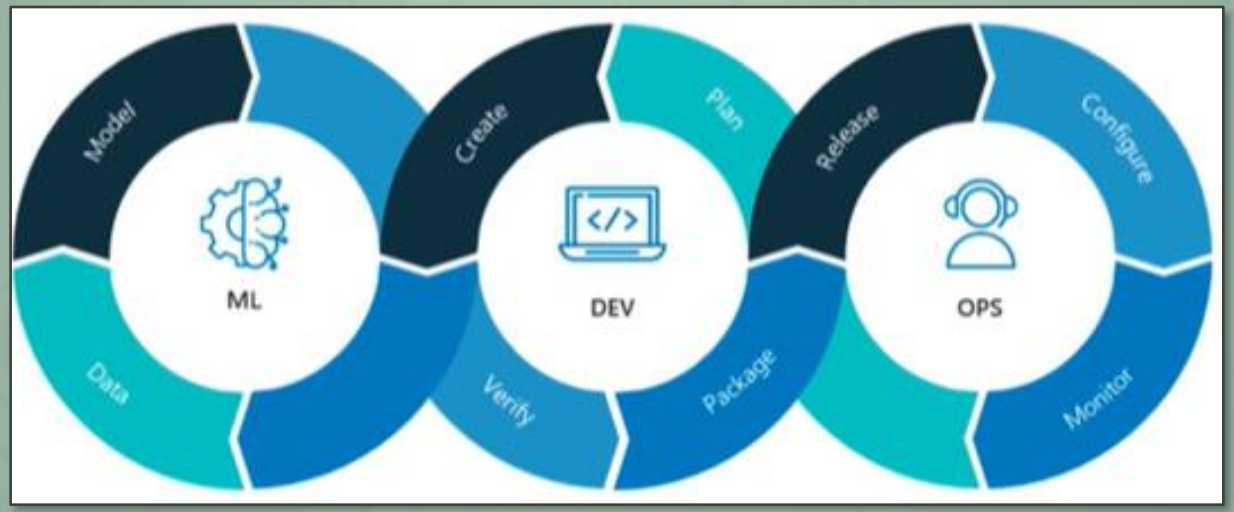


Fields of Machine Learning



What is MLOps or Machine Learning Operations?

…it refers to practices for collaboration and communication between data scientists, developers and operations professionals to help manage production ML (or AI) lifecycle, i.e. take machine learning models to production, maintaining and monitoring them



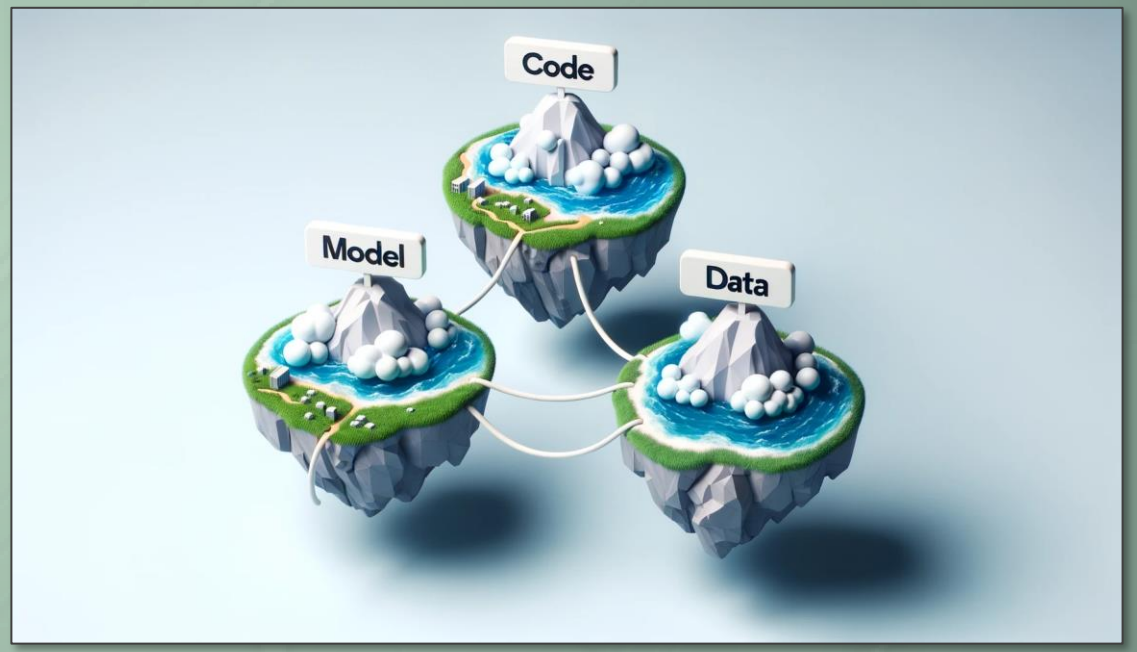
Aspect	Function
Automation	Streamlining process from data collection to model deployment.
Reproducibility	Ensuring consistent environments and results across ML lifecycle.
Monitoring	Tracking model performance and data quality in production.
Collaboration	Facilitating seamless interactions between cross-functional teams.
Lifecycle Management	Managing end-to-end workflow from data preparation to model serving and monitoring.

Changing anything changes everything

In MLOps, every alteration in data, code, or environment can have far-reaching effects across the entire machine learning pipeline.

MLOps emphasizes the need for rigorous versioning, testing, and monitoring to manage and mitigate the cascading impacts of changes.

Change	Potential Ripple Effects
Data Update	▪ Model retraining ▪ altered predictions ▪ need for new validation
Code Revision	▪ Adjustments in data processing ▪ potential shifts in model performance
Environment Shift	▪ Reproducibility challenges ▪ deployment inconsistencies ▪ varied outcomes



ML management vs. Software engineering

Traditional Software Engineering	ML Model Management
Focus: Code quality & functionality	Focus: Data quality & model accuracy
Change Source: Primarily code updates	Change Source: Code, data, or model updates
Validation: Functional tests	Validation: Model evaluation metrics
Deployment: Version control, CI/CD	Deployment: Version control for code, data, & models
Behavior: Deterministic	Behavior: Probabilistic

MLOps as evolution of traditional DevOps

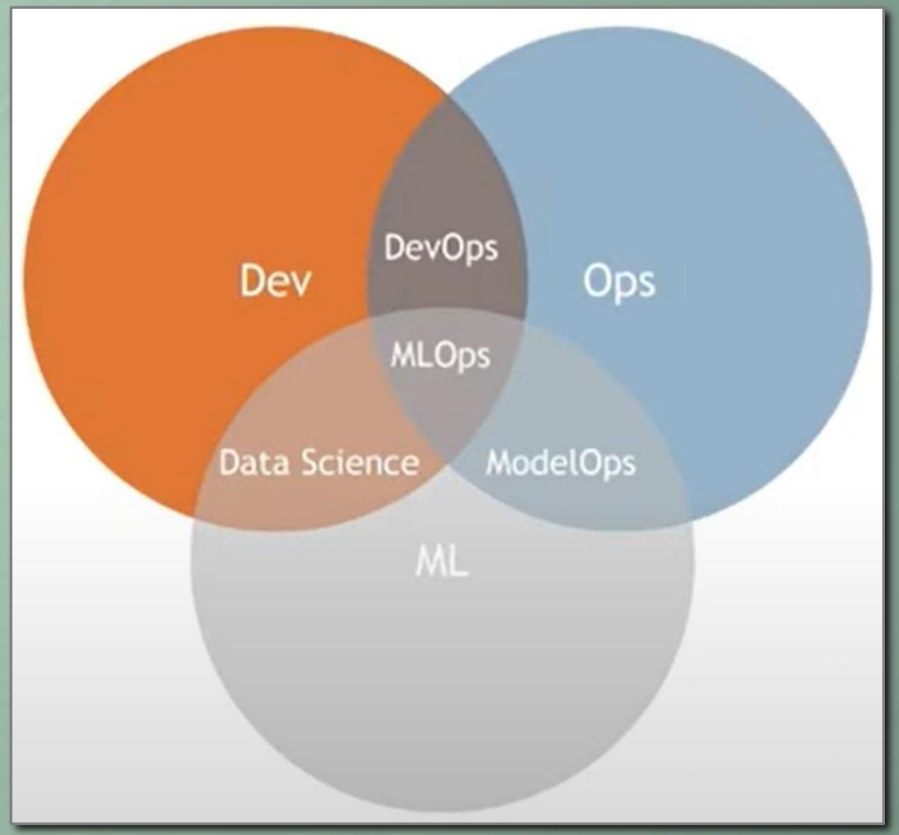
MLOps extends the principles of DevOps to address the unique challenges of machine learning, ensuring that models are as robust and reliable as the infrastructure they run on.

DevOps:
Integrates development and operations for software delivery

- Focuses on code deployment and infrastructure management.
- Emphasizes speed, efficiency, and scalability.
- Tools: git/GitHub, Docker, Kubernetes.

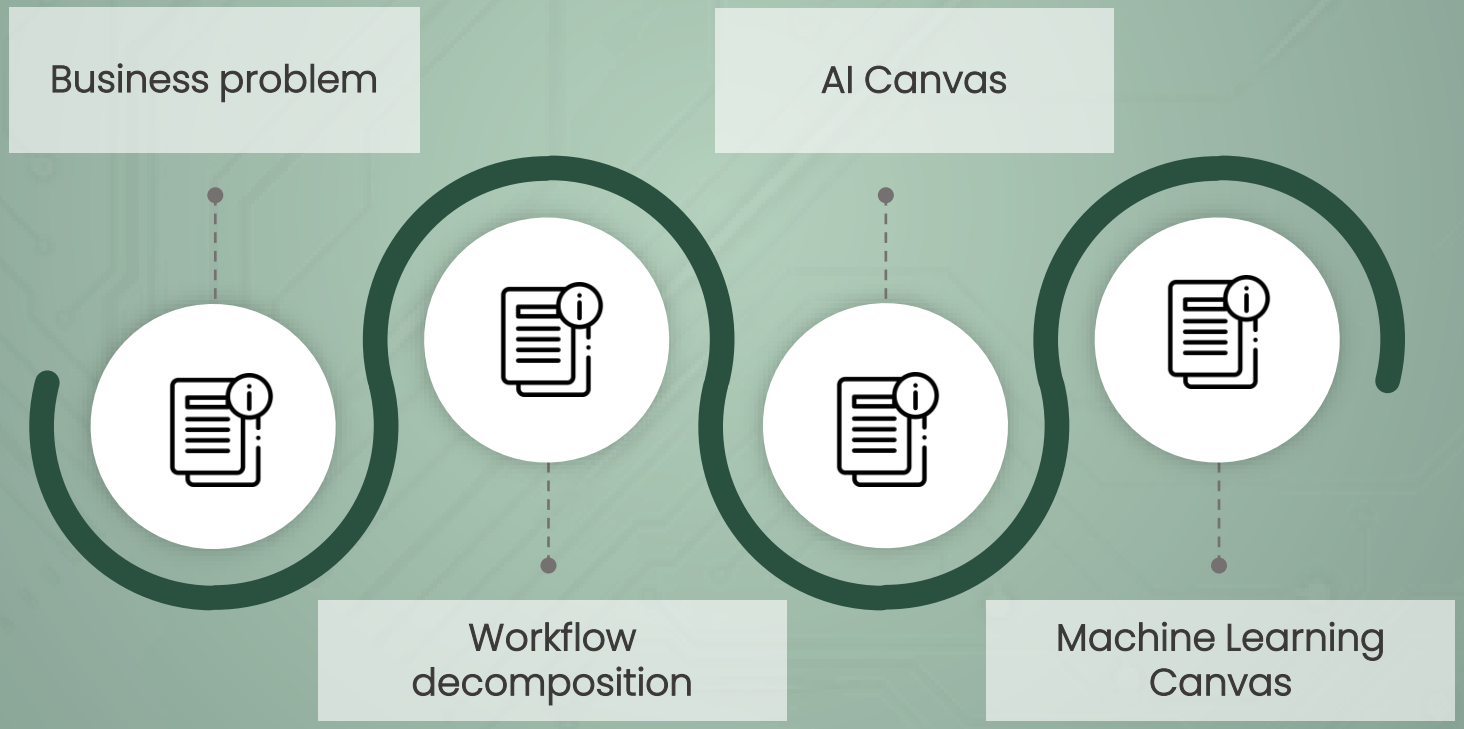
MLOps:
Enhances DevOps with machine learning specifics

- Encompasses model development, deployment, and monitoring.
- Adds layers for data versioning, model validation, experiment tracking.
- Tools: MLflow, Kubeflow, TFX.



Business Problems requiring ML

Key to any project, especially in ML, is a deep understanding of the business problem.



Understanding the Business Problem

...is crucial for successful ML project outcomes, driving the creation of accurate models and defining the threshold for success.

Initial Step

- Conduct a thorough study of the business problem.
- Define clear requirements to shape the ML project.

Translation to ML

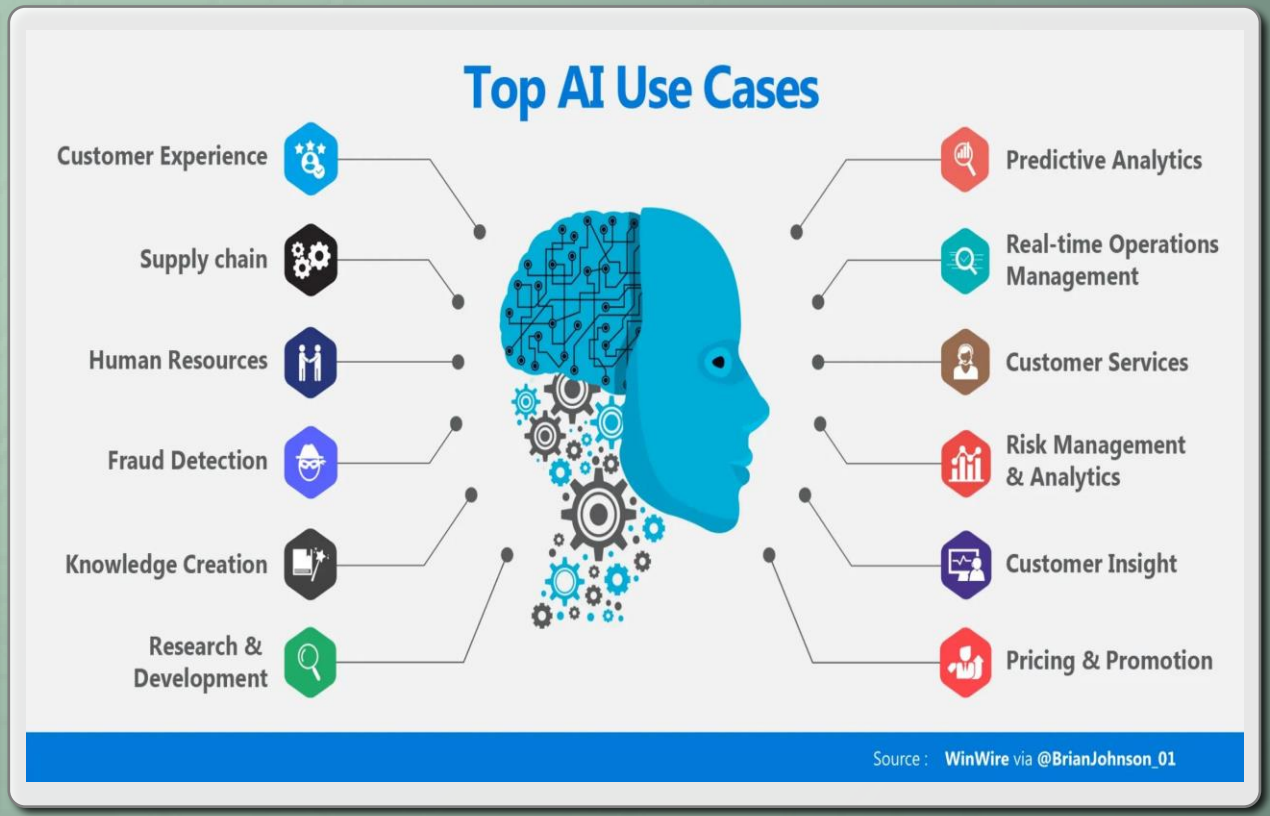
- Convert business requirements into actionable model objectives.
- Determine the desired model outputs.

Specifying Success Criteria

- Establish benchmarks for acceptable performance.
- Identify potential errors and their implications.

Feasibility Determination

- Assess cost of inaccuracies: "How costly are wrong predictions?"
- Calculate the impact of prediction errors on the business.



Work Flow Decomposition

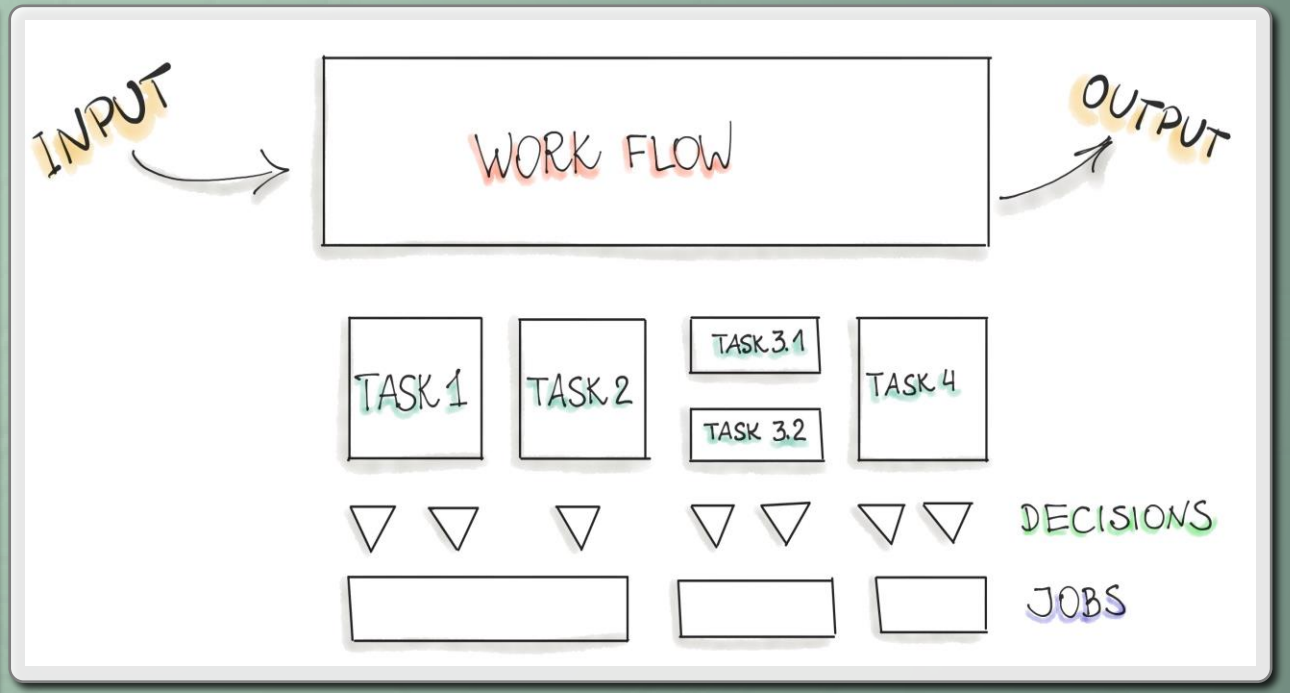
Decomposing workflows and assessing ROI are critical for strategic AI/ML implementation, ensuring impactful and cost-effective enhancements.

Task Decomposition

- Break down business process into elemental tasks.
- Determine integration points for ML models.

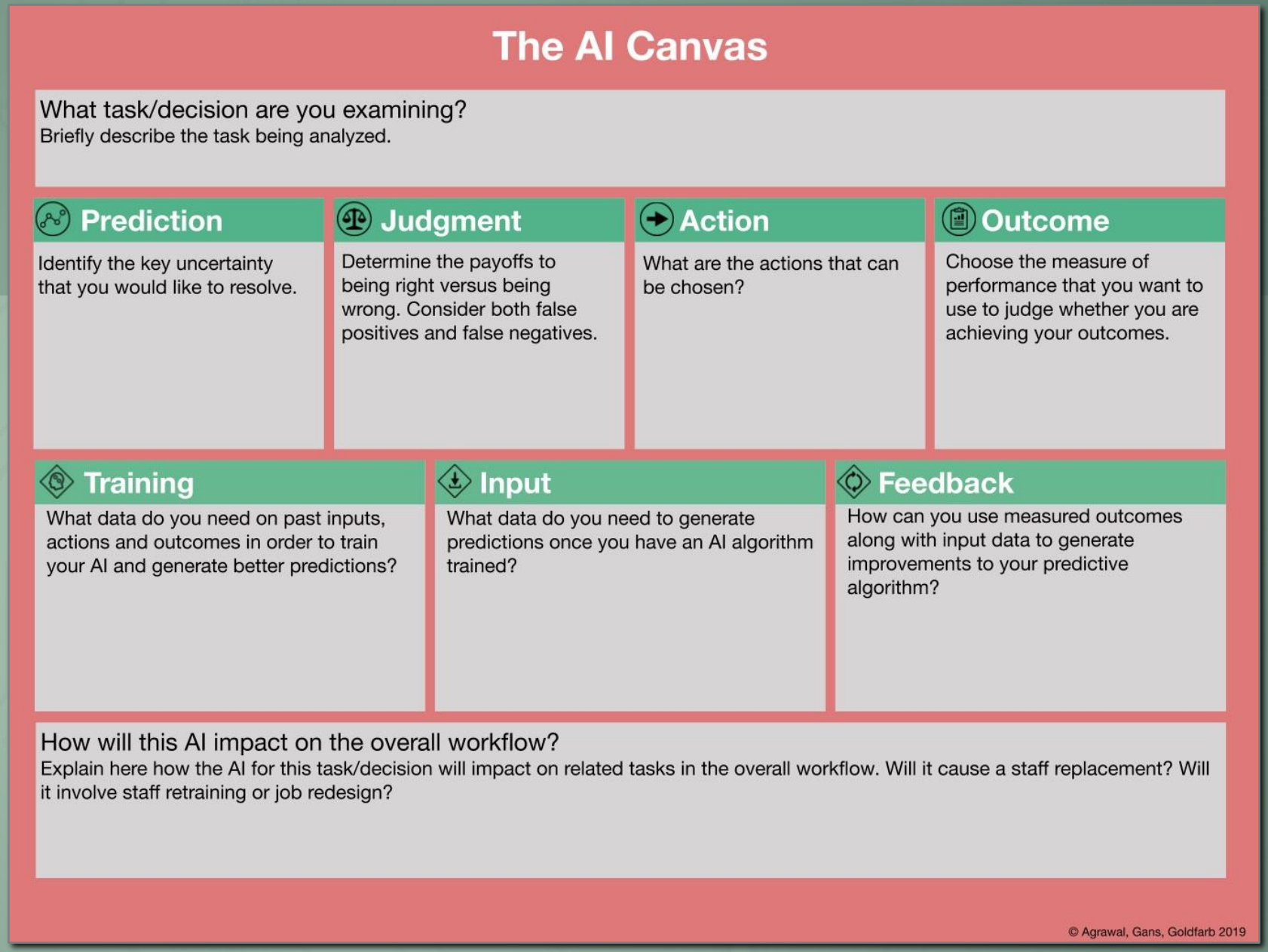
Implementation Steps

1. **Process Identification:**
   Locate process suitable for AI/ML enhancement.
2. **Task Graphing:**
   Create directed graph detailing sequence of tasks.
3. **Human Element Analysis:**
   Pinpoint tasks where human intervention can be substituted with AI/ML.
4. **ROI (return on investment) Estimation:**
   Calculate potential ROI for automating each task with AI/ML.
5. **Prioritization:**
   Rank tasks by ROI to guide AI/ML implementation strategy.



AI Canvas

… is aid for contemplating, building, assessing AI tools



Machine Learning Canvas

… is strategic tool to articulate vision and specifics of ML project, ensuring that each component aligns with overall value proposition.

Objective Identification

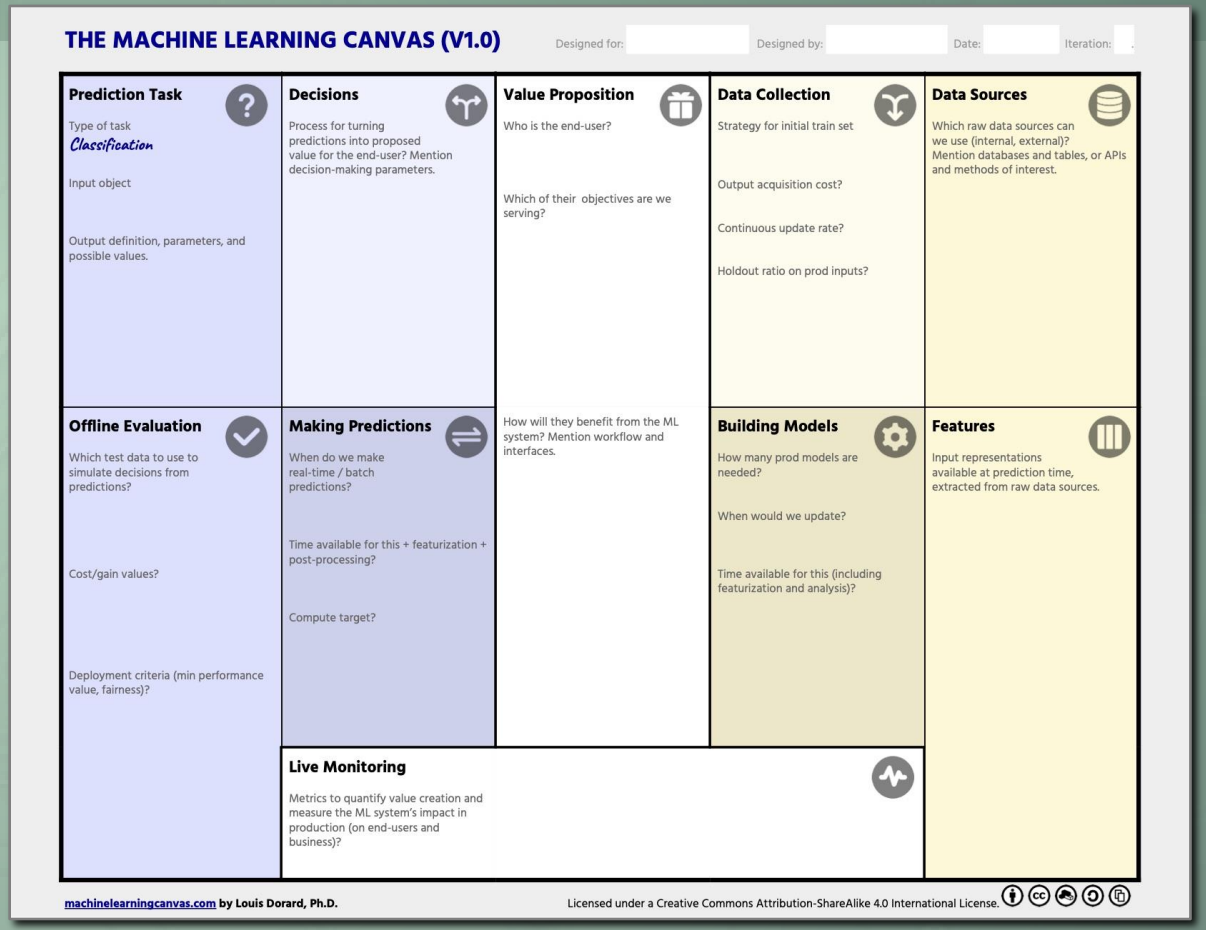
- Define what ML system aims to achieve for its users.
- Align business goals with ML tasks.

Value Proposition

- What problem does it solve?
- Why is it important?
- Who are the end-users?
- What value is delivered to the end-user?
- How will end-users utilize the outputs/predictions?

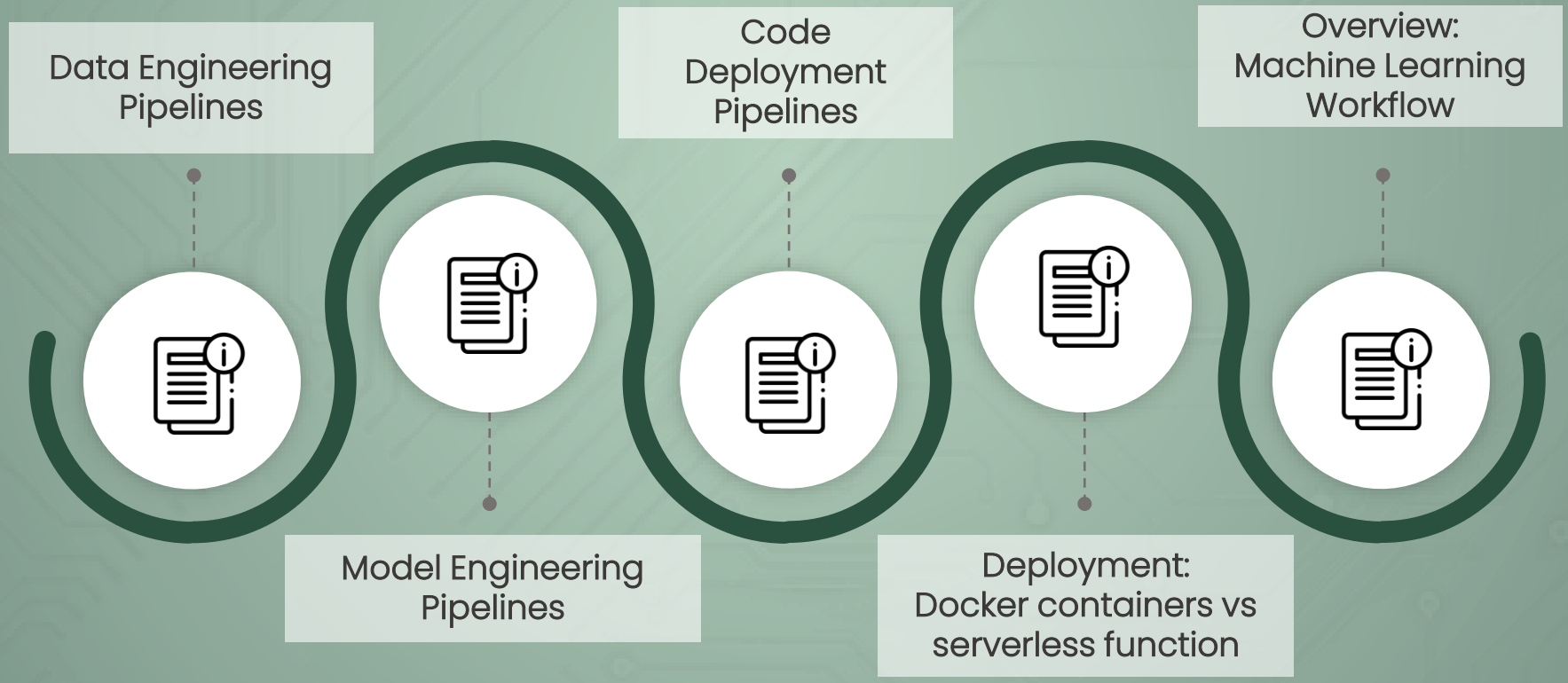
Canvas Categories

- Learning: Detailing approach for model training.
- Prediction: Outlining how / when predictions will be made.
- Evaluation: Defining evaluation methods, performance metrics.



Overview of Machine Learning engineering

Three levels of ML software: Data, Models, Code



Data Engineering Pipelines

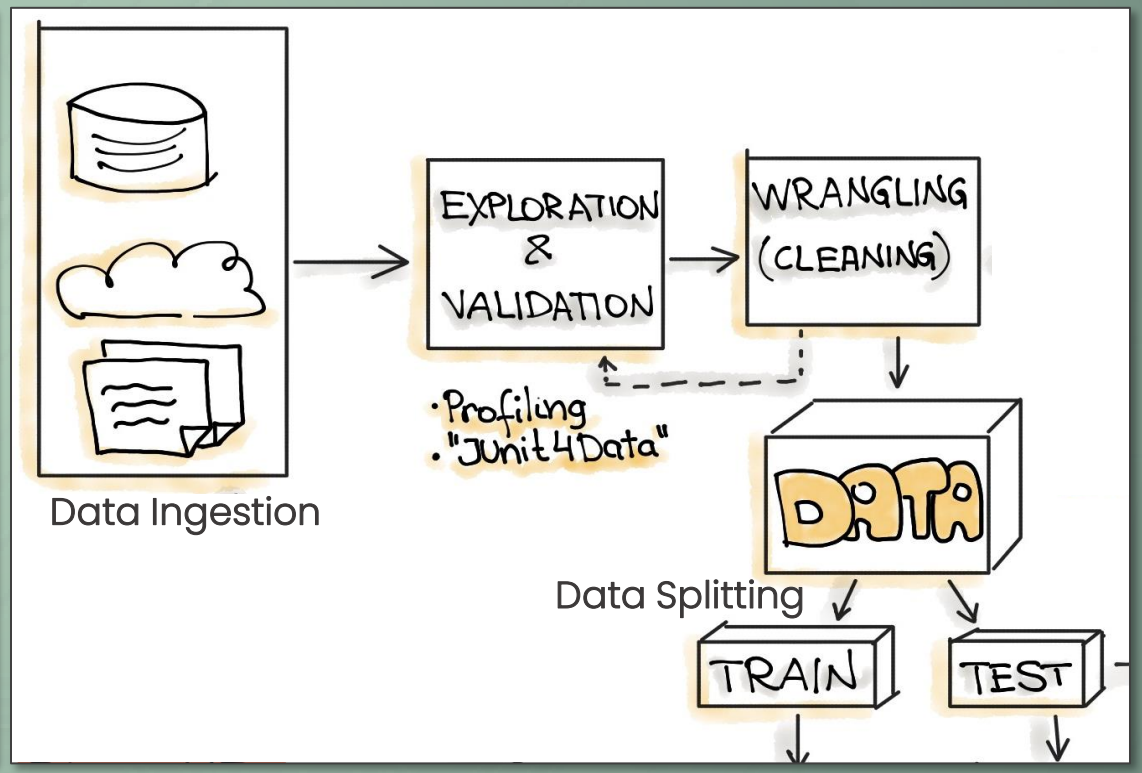
Efficient data engineering pipelines are crucial for curating the high-quality data needed for robust ML model performance.

Foundational Role of Data in ML

- Quality data is key to the success of ML models.
- "Garbage In, Garbage Out": model quality is contingent on data quality.
- Data used in training profoundly influences production performance.

Pipeline Stages

1. Data Ingestion:
   Collecting and assembling data from various sources.
2. Exploration and Validation:
   Analyzing data to understand its structure and integrity.
3. Data Wrangling (Cleaning):
   Improving data quality for optimal model training.
4. Data Splitting:
   Segregating data into distinct sets for training and testing algorithms.

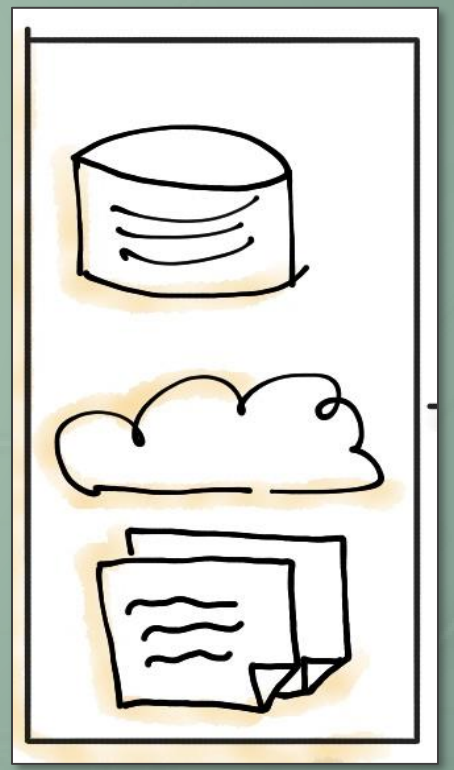


Data Engineering Pipelines: Data Ingestion

Data ingestion is the critical first step in ML workflows, laying the groundwork for effective model training and avoiding biases.

Key Actions in Data Ingestion

- Data Sources Identification:
  Document origins for traceability.
- Space Estimation:
  Assess storage requirements.
- Obtaining Data:
  Acquire and format data for usability.
- Back up Data:
  Protect the integrity of the original data.
- Privacy Compliance:
  Adhere to regulations like DSVGO (EU regulation).
- Metadata Catalog:
  Record dataset metadata for future reference.



Data Engineering Pipelines: Exploration and Validation

… are crucial for ensuring data integrity

Overview of Exploration

- Perform data profiling to understand content and structure.
- Metadata output includes statistical measures like max, min, average values.

Validation Process

- Document metadata for each attribute
- Validate data attributes for consistency and correctness (e.g., addresses, postal codes)
- Conduct data validation to check quality, using error detection functions.
- Create visual representations to understand value distributions.
- Analyze correlations to gain insights into data relationships.

Data Engineering Pipelines: Data Wrangling (Cleaning)

Enhance data quality by reformatting and restructuring to meet the schema requirements.

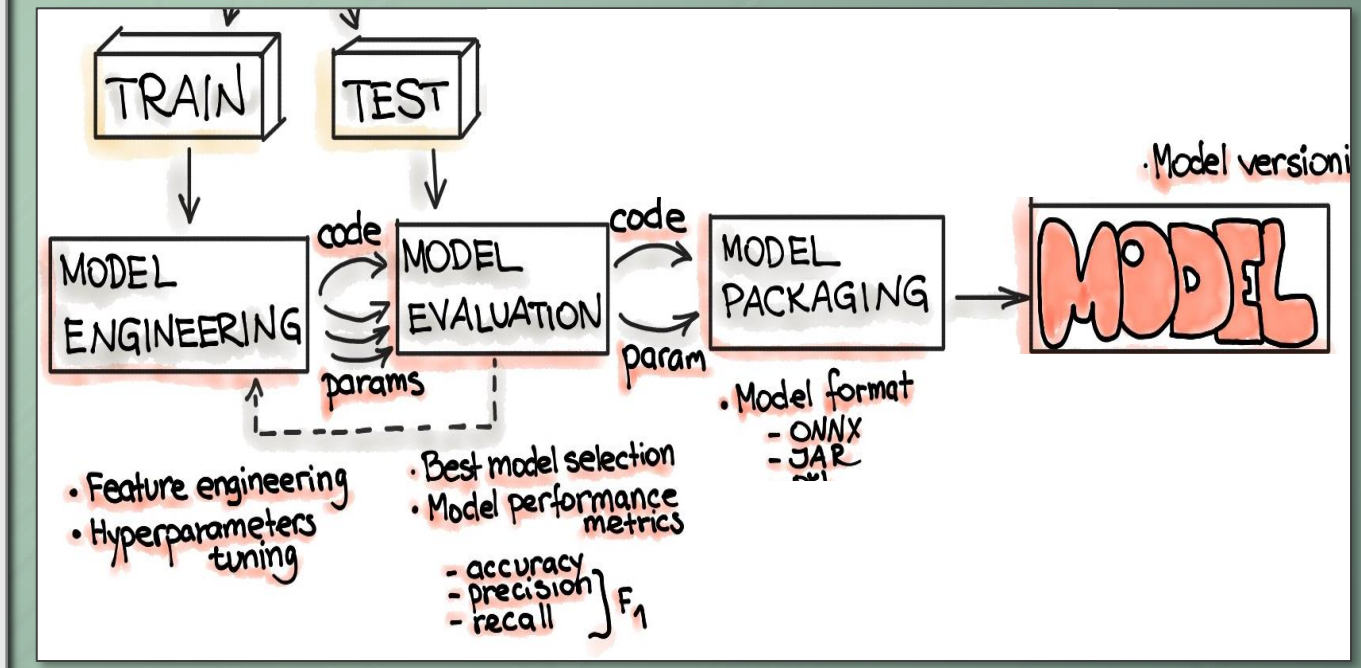
Key Wrangling Actions

1. Transformations:
   Determine beneficial data transformations.
2. Outliers:
   Decide on fixing or removing statistical outliers.
3. Missing Values:
   Impute missing entries with zero, mean, median, or remove affected rows/columns.
4. Irrelevant Data:
   Eliminate features that don't contribute to the analysis.
5. Restructuring Operations:
   - Reorder, combine, or create new record fields.
   - Filter datasets and adjust granularity through aggregations and pivots.

Model Engineering Pipelines

The heart of the ML workflow is developing the ML model through a sequence of key operations

1. Model Training:
   - Applying algorithms to data
   - Feature engineering:
     Crafting features to improve model performance.
   - Hyperparameter tuning:
     Optimizing model parameters for better results.
2. Model Evaluation:
   Validating the model against business objectives.
3. Model Testing:
   Estimating generalization error with a hold-back test set.
4. Model Packaging:
   Exporting the model for integration into applications.



Model Engineering Pipelines: Model serving patterns

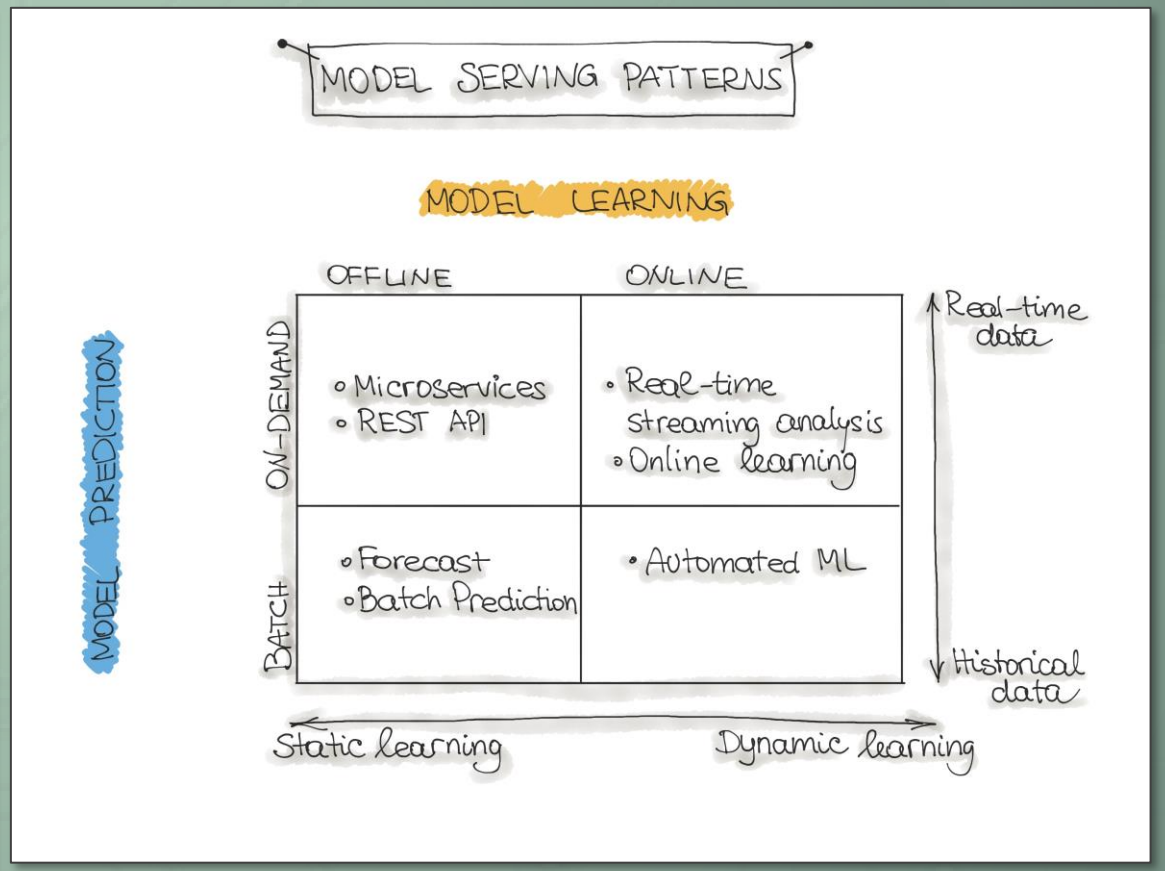
Understanding the nuances between different ML workflow patterns is crucial for designing a system that aligns with business needs and data nature.

ML Model Training

- Offline Learning:
  - Trained on historical data.
  - Model remains unchanged post-deployment until retraining.
  - Watch for 'model decay'.
- Online Learning:
  - Continuously retrained with new incoming data.
  - Ideal for time-series data like sensors or stock trading.

ML Model Prediction

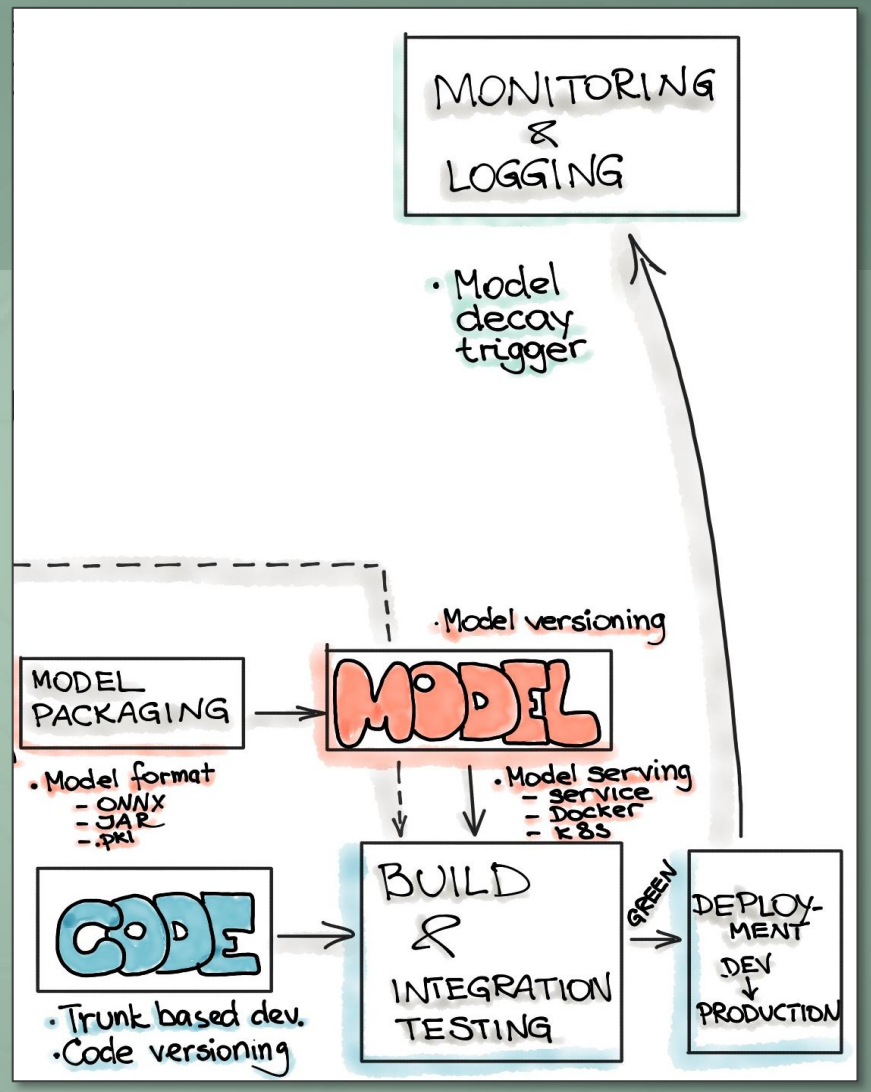
- Batch Predictions:
  - Model predicts in sets from historical data.
  - Suitable for non-time-critical predictions.
- Real-time Predictions:
  - Model predicts on-the-fly with live data.
  - Necessary for time-sensitive decision-making.



Code Deployment Pipelines

Integrating ML into Business Applications

- **Model Serving:**
  Deploying the model to production for inference.
- **Model Performance Monitoring:**
  Observing model behavior and performance in real-world scenarios.
- **Model Performance Logging:**
  Recording each inference for audit and analysis.

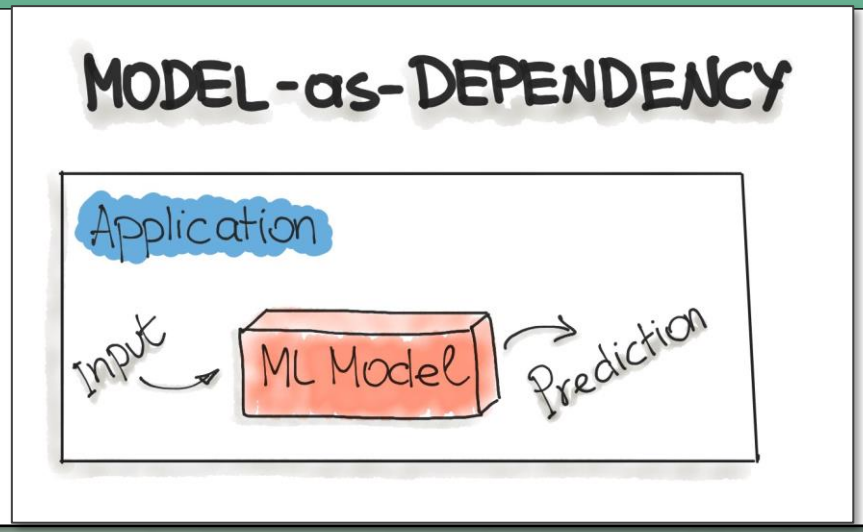


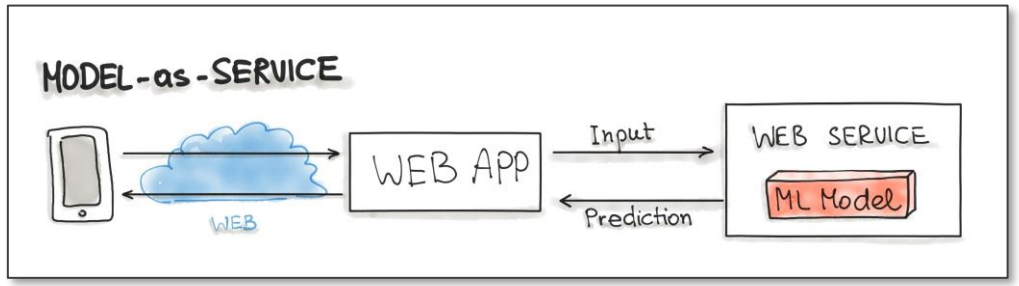
Code Deployment Pipelines: Model serving patterns

Integrating ML into Business Applications

	ML Model		
Service & Versioning	Together with the consuming application	Independent from the consuming application	
Compile/ Runtime Availabilty	Build & runtime available	Available remotely through REST API/RPC	Available on demand
Serving Patterns	"Model-as-Dependency"	"Model-as-Service"	▪ "Precompute" ▪ "Model on Demand"
	Hybrid Model Serving (Federated Learning)		

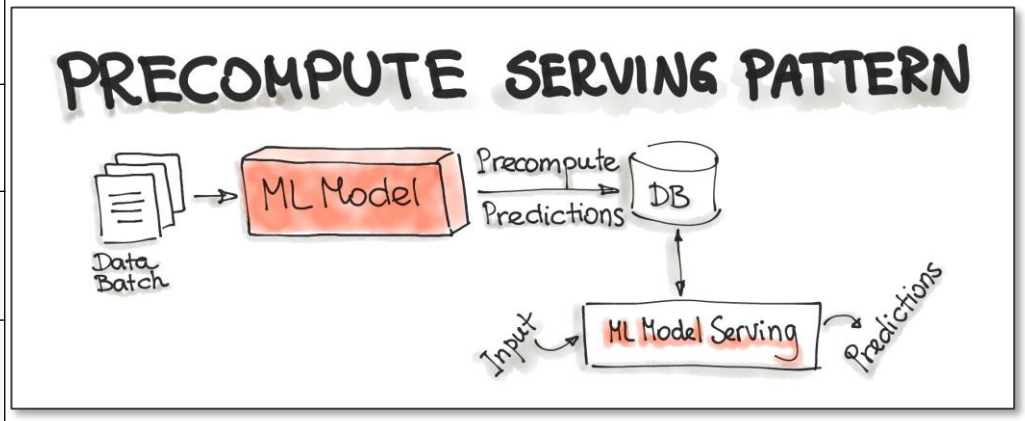
Code Deployment Pipelines: Model serving patterns

	ML Model	
Definition	Integrating ML models as dependencies in applications.	
Implementation	Applications invoke prediction methods directly from the model.	
Use Cases	Commonly used in Forecasting scenarios.	
Benefits	Simplifies deployment, direct use within the application.	

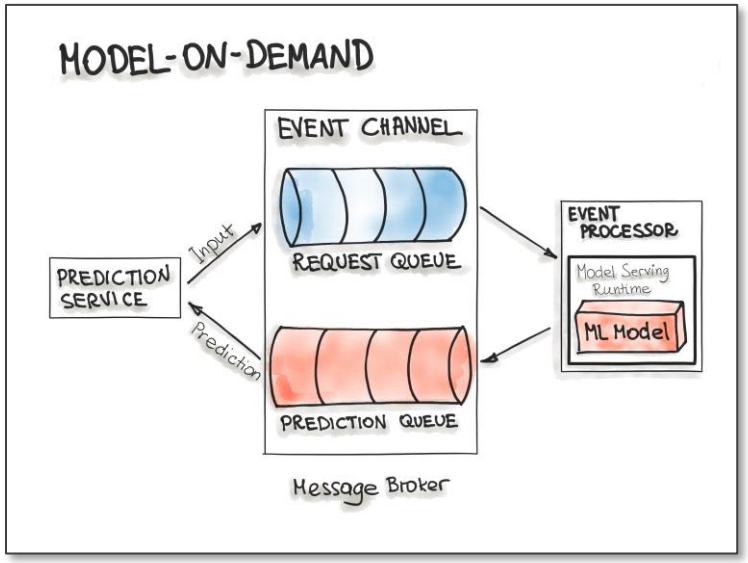
	ML Model	
Definition	Encapsulating ML models as standalone services.	
Implementation	Models are accessed via REST API or gRPC service.	
Use Cases	Ideal for Forecast, Web Service, and Online Learning workflows.	
Benefits	Decouples model lifecycle from application, scalable.	

Code Deployment Pipelines: Model serving patterns

	ML Model
Definition	Precomputing predictions with a trained ML model.
Implementation	Predictions are stored in database, queried upon request.
Use Cases	Suited to Forecast workflows where predictions are not time-sensitive.
Benefits	Predictions are ready on demand, reduces compute load during request time.



	ML Model
Definition	An ML model deployed as an independent service, accessible at runtime.
Implementation	Employs message-broker architecture with a broker for event channels and an event processor for prediction execution.
Use Cases	Best for real-time analytics and dynamic data environments.
Benefits	▪ Scalable to demand fluctuations. ▪ Independent model updates. ▪ Quick prediction delivery.



Deployment Strategies for ML Models

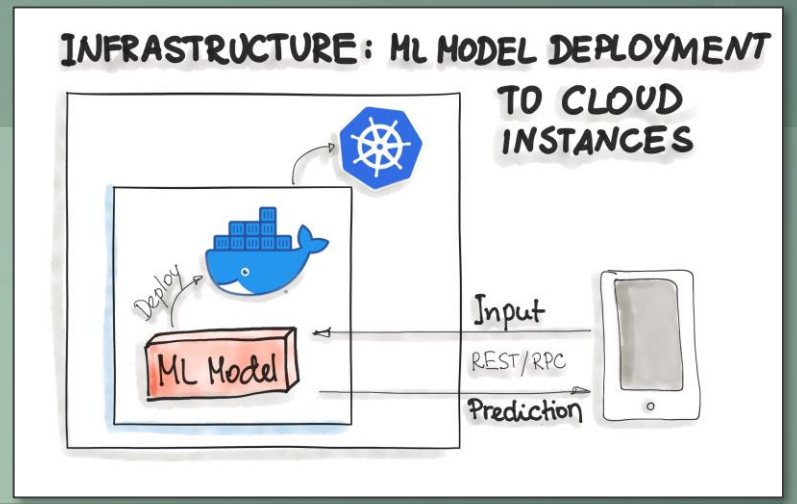
Docker container vs. (serverless) lambda functions

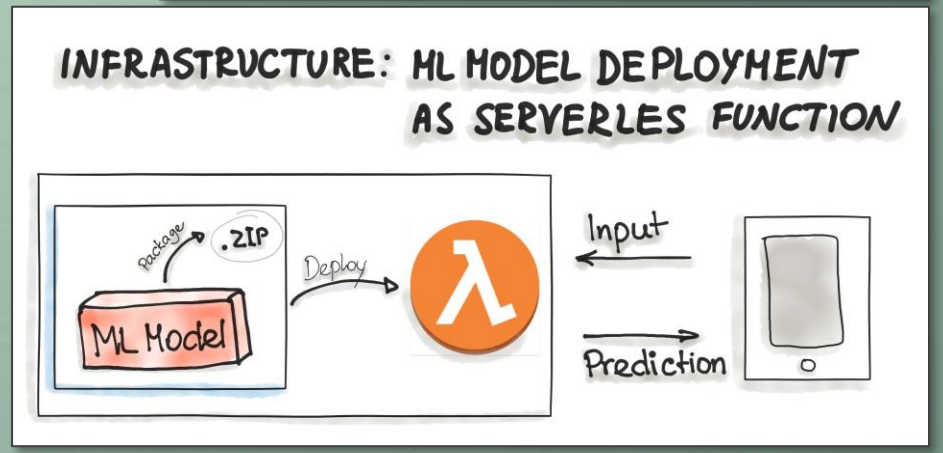
Containerization

- Docker Containers:
  Package ML models and dependencies into containers.
- Orchestration:
  Managed by AWS ECS/Fargate or Kubernetes
- Access:
  Models served via REST API, frontend by e.g. FastAPI

Serverless Deployment

- Cloud ML Platforms:
  AWS Sagemaker, Google Cloud AI, Azure ML Studio
- Serverless Functions:
  Package code into .zip for deployment via AWS Lambda, Azure Functions, Google Cloud Functions.
- Considerations:
  Be mindful of constraints like artifact size. Calculation time not more than 15 minutes at AWS Lambda

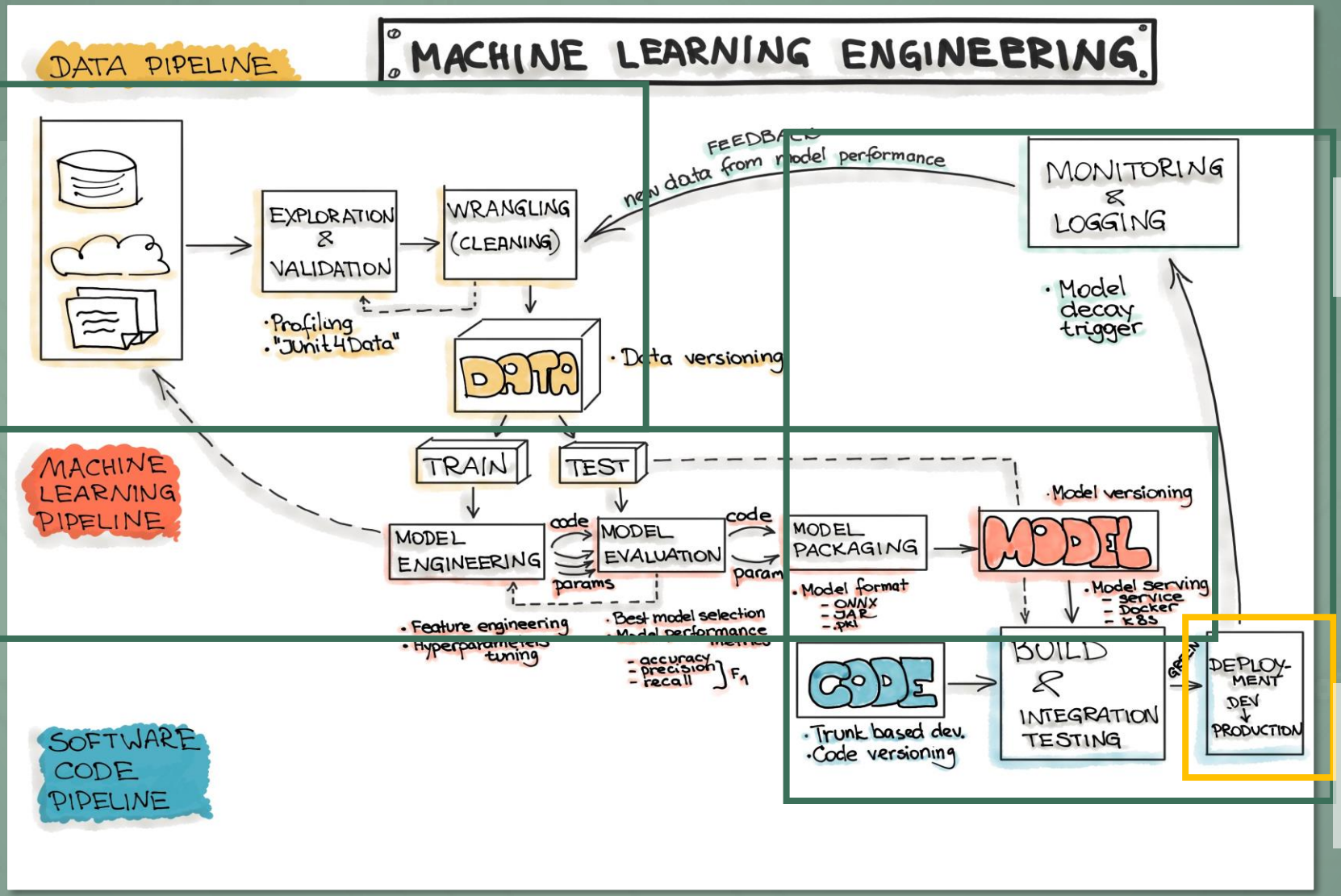




Overview of Machine Learning Engineering

Slides:
Data Engineering
Pipelines

Slides:
Model Engineering
Pipelines

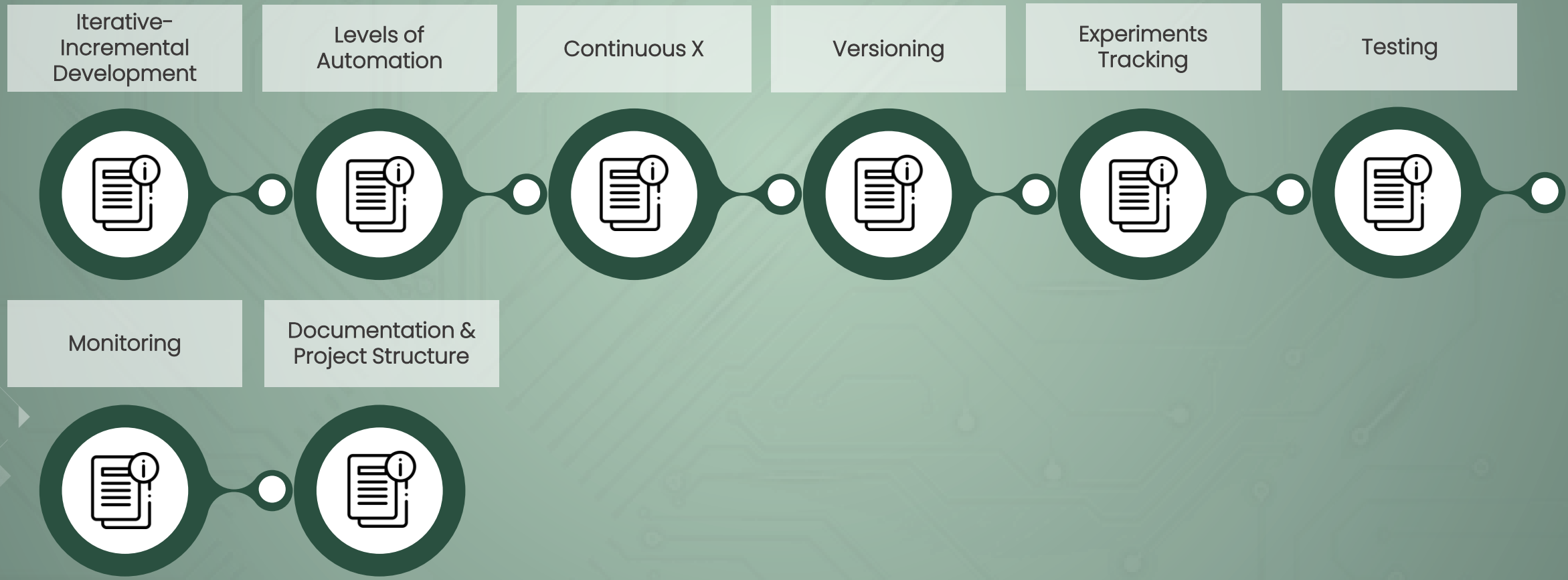


Code
Deployment
Pipelines

Deployment:
Docker containers
vs serverless
function

MLOps Principles

… are designed to integrate ML development seamlessly into software production, ensuring efficient, reliable, and timely delivery of AI capabilities.



Iterative-Incremental Process

The MLOps process is an agile and iterative journey from design to deployment, ensuring that ML applications meet both business and technical requirements.

Designing ML-Powered Applications

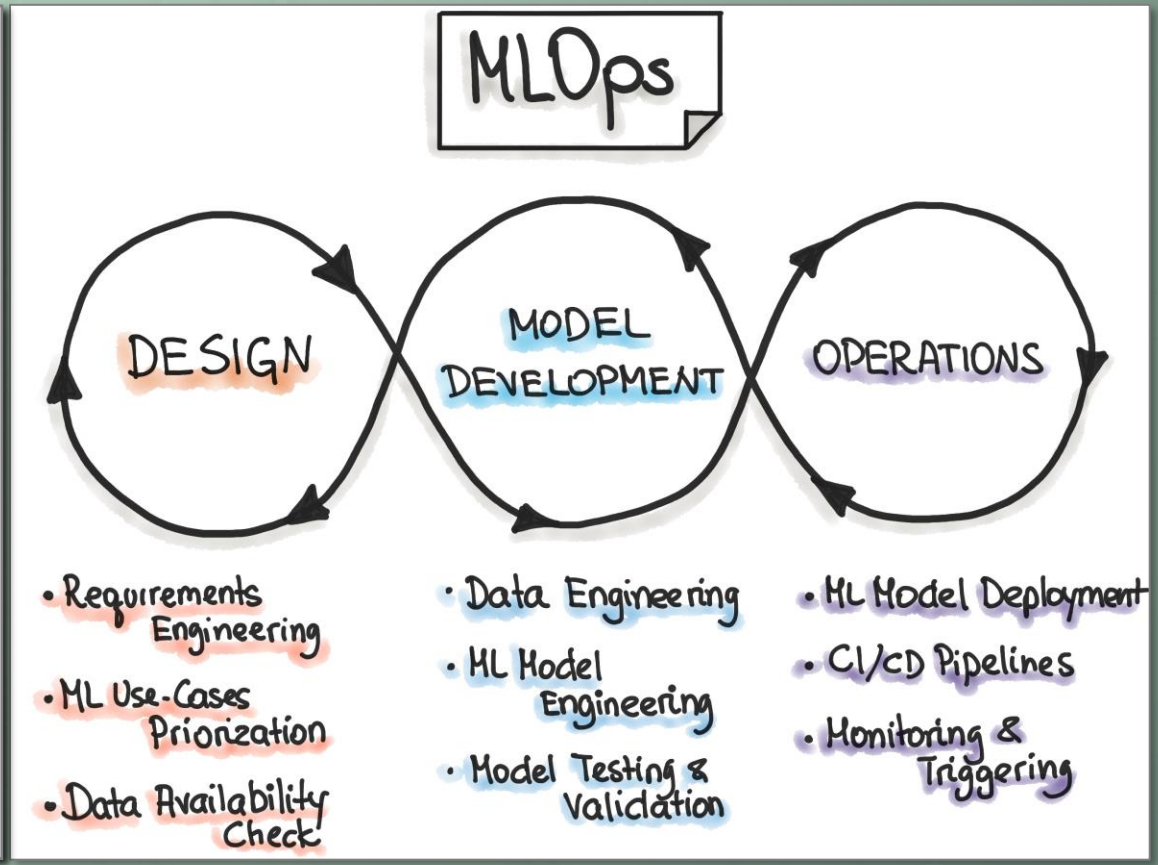
- Focus on business and data understanding.
- Identify end-users and design ML solutions.
- Assess project development pathways.
- Tackle one use case at time for optimal focus, resources allocation.

ML Experimentation and Development

- Validate ML applicability through Proof-of-Concept models.
- Iterative process of algorithm selection, data engineering, and model engineering.
- Goal: Deliver a stable, production-ready ML model.

ML Operations

- Implement DevOps practices:
  testing, versioning, continuous delivery monitoring, etc.
- Deploy ML models in production environments.



Levels of Automation

Strategic automation across MLOps processes enhances the efficiency, reliability, and speed of ML model development and deployment.

Automation Objectives

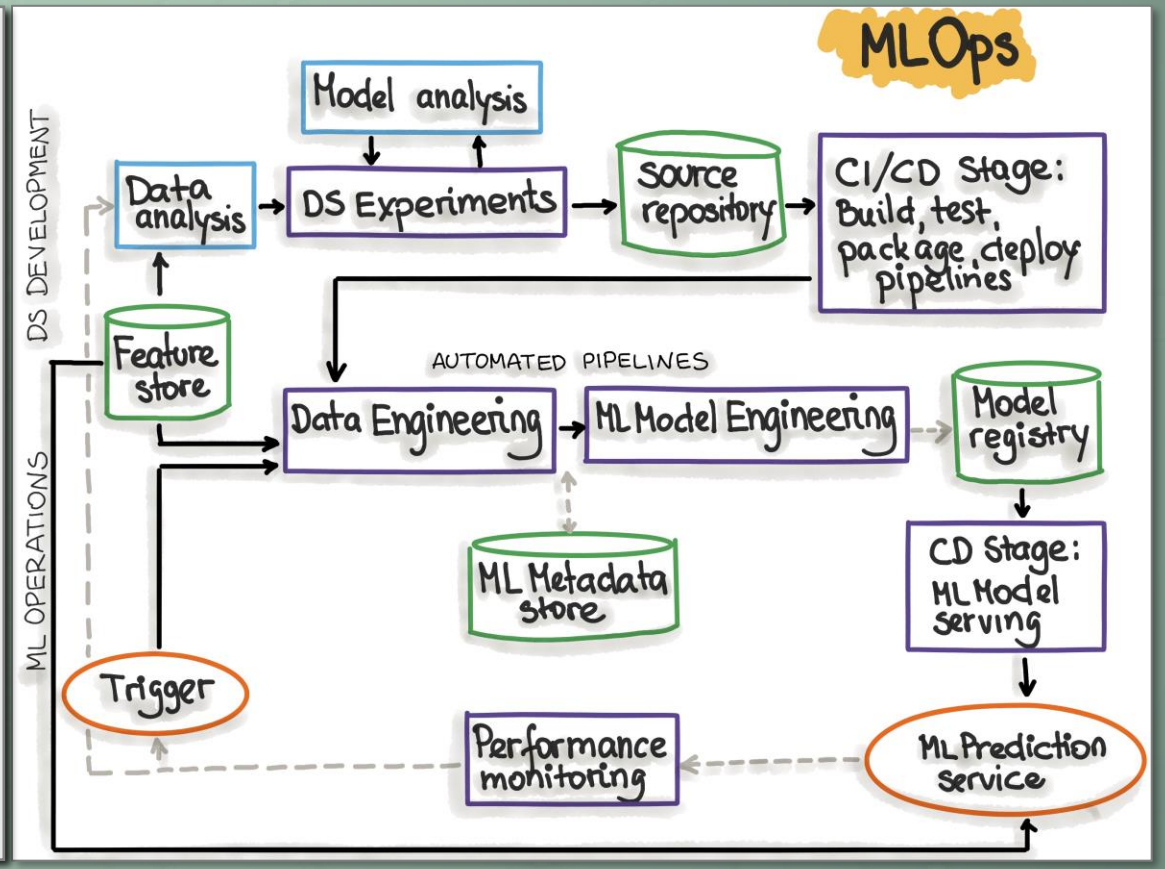
- Goal: Integrate ML model deployment seamlessly into core systems or as standalone services.
- Achieve full workflow automation without manual steps.
- Early problem detection, resolution und continuous improvement

Triggers for Automation

- Scheduled events, monitoring alerts, and updates to data, training code, or application code.

Levels of Automation

1. Manual Process: manual execution of all pipeline steps using tools like Jupyter Notebooks.
2. ML Pipeline Automation: Automatic model training with continuous retraining triggers, including validation steps.
3. CI/CD Pipeline Automation: Full integration of CI/CD for reliable ML deployments, automating build, test, and deployment processes.



Continuous X

The 'Continuous' in MLOps: Integration, Delivery, Training, Monitoring

Understanding ML Assets

- Identify ML models, parameters, scripts, data, their versions and dependencies.
- **Goal:** Stable deployment of ML artifacts to services or infrastructure.

Continuous Practices

- Continuous Integration (CI):
  Test and validate not just code but also data and models.
- Continuous Delivery (CD):
  Deploy ML training pipelines: can also update prediction services automatically.
- Continuous Training (CT):
  Unique to ML, involving automatic retraining and redeployment of models.
- Continuous Monitoring (CM):
  Oversee production data, model performance, aligning with key business metrics.

Versioning

… is crucial for the systematic tracking, auditing, and management of machine learning models and associated data

Triggers for Version Updates

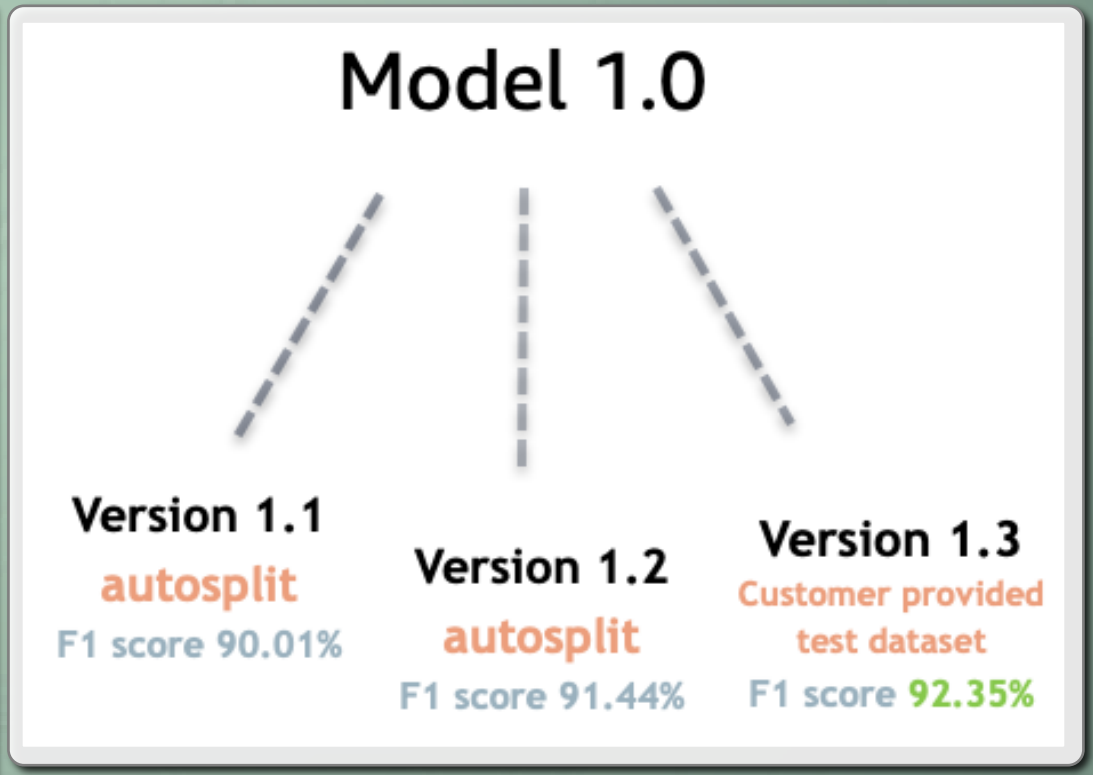
- Introduction of new training data.
- Application of innovative training methods.
- Addressing model performance decay (**model drift**).
- Deployment adjustments for new application contexts.
- Compliance with legal and regulatory audits.

Data Governance

- Ensuring data consistency across systems.
- Upholding data immutability and ownership.

Best Practices

- Conduct code reviews for all ML specifications.
- Use Version Control Systems (VCS) for full auditability, reproducibility.



F1 score: measure of model's accuracy in binary classification problem

Experiments Tracking

… facilitates selecting the best-performing models for production

Tracking Scenarios

- Implement parallel experimentation to optimize model training
- Use Git branches for individual experiments.
- Evaluate and compare models based on key performance metrics.

Tools for Tracking

- DVC: Data Version Control (https://dvc.org)
  - Git extension tailored for ML projects.
  - Facilitates branch management and versioning of ML models.
- Weights & Biases (https://wandb.ai)
  - Automates tracking of hyperparameters and performance metrics.
  - Streamlines the experiment comparison process.

Multifaceted Testing for Machine Learning Integrity

… categorized into testing for features & data, model development, and ML infrastructure

ML Infrastructure Tests

- Ensure reproducibility and determinism in training.
- Conduct stress testing and crash tests for robustness.
- Implement integration tests across the full ML pipeline.

Features and Data Tests

- Data validation and feature importance assessments.
- Compliance checks and unit testing for feature creation.

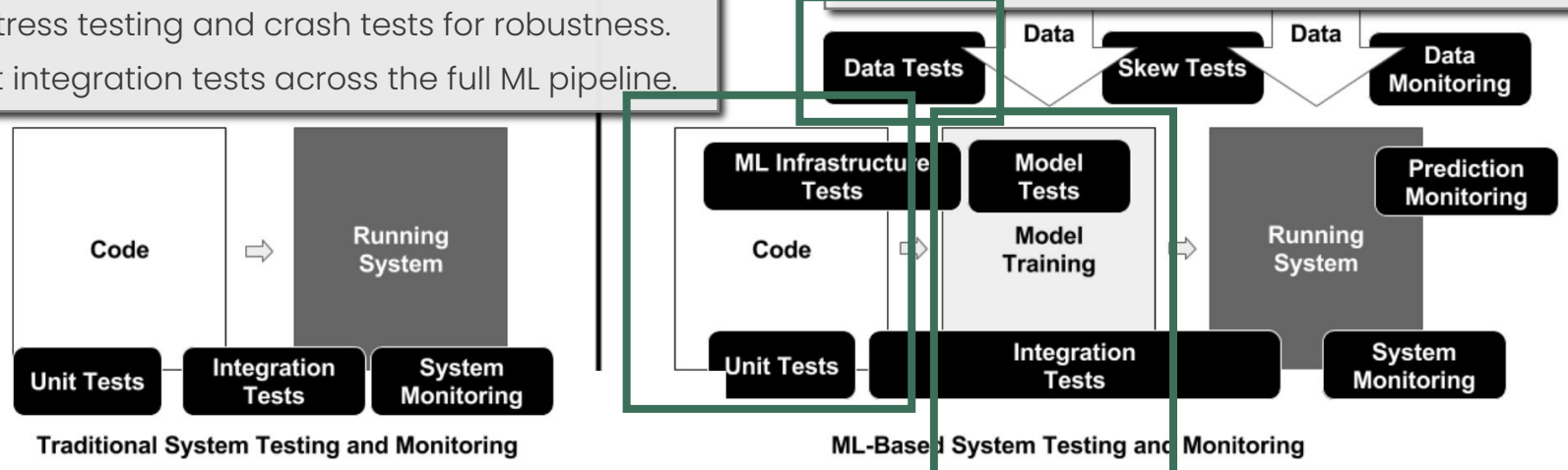


Figure 1. **ML Systems Require Extensive Testing and Monitoring.** The key consideration is that unlike a manually coded system (left), ML-based system behavior is not easily specified in advance. This behavior depends on dynamic qualities of the data, and on various model configuration choices.

Model Development Tests

- Align ML loss metrics with business impact measures.
- Perform staleness checks and bias/fairness evaluations.
- Utilize A/B testing and additional test sets for model validation.

Effective Monitoring for ML Model Performance

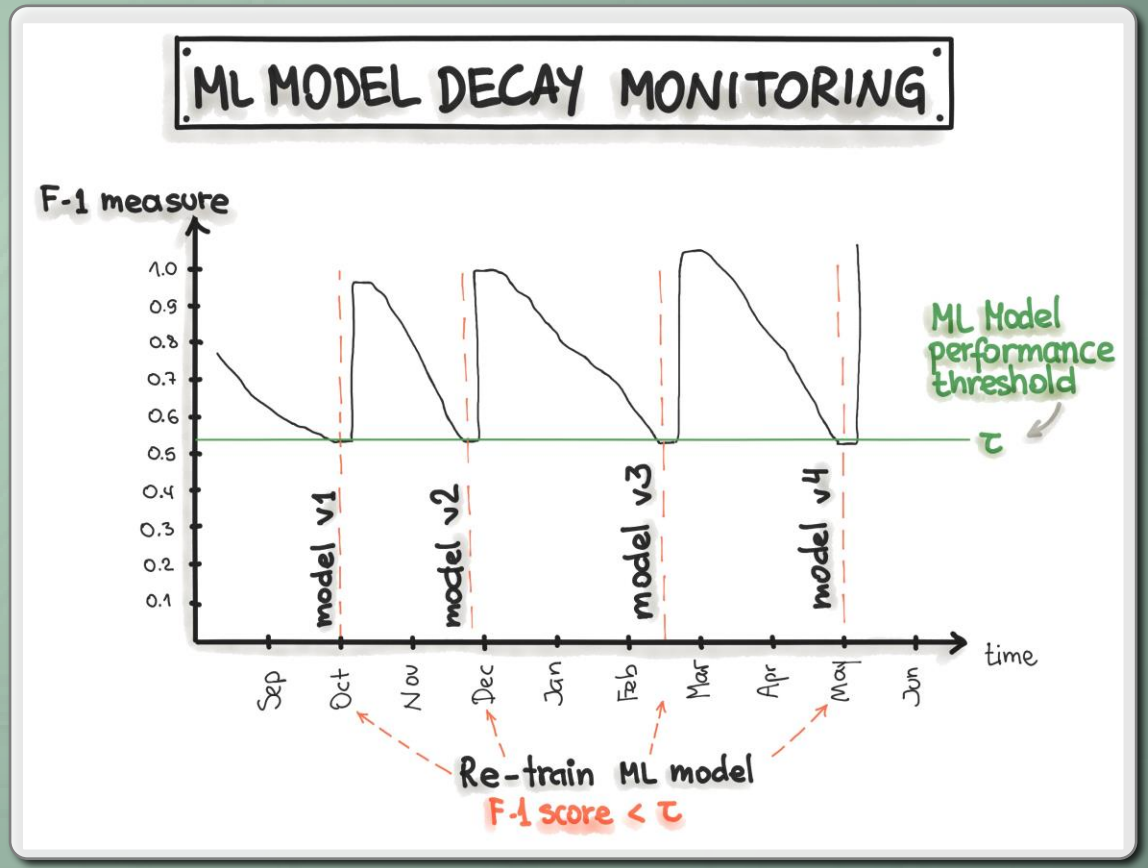
… is about maintaining accuracy and efficiency of ML models in production, ensuring timely updates and optimization

Metrics to Track

- Precision, recall, and F1-score over time (possible delay / model drift?)
- Model version performance.
- Computational resource usage

Key Monitoring Actions

- Adjust alert thresholds for meaningful notifications.
- Log serving traffic and compare with training data statistics.
- Gather system metrics to inform resource allocation and costs.
- Regularly update feature generation processes.
- Continuously evaluate prediction quality and bias.



Documentation & Project Structure

… facilitates maintaining clarity, consistency and efficiency in MLOps.

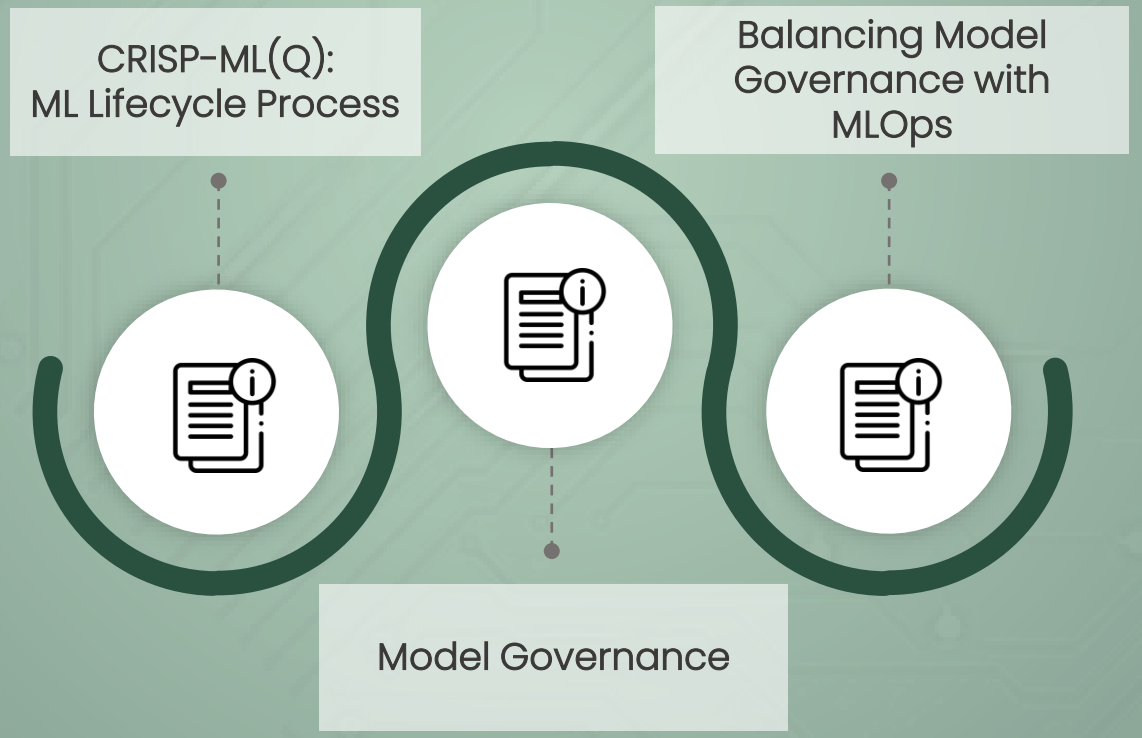
MLOps Best Practices	Documentation	Project Structure
Data	▪ Data sources ▪ Decisions, how/where to get data ▪ Labelling methods	▪ Data folder for raw and processed data ▪ Folder for data engineering pipeline ▪ Test folder for data engineering methods
ML Model	▪ Model selection criteria ▪ Design of experiments ▪ Model pseudo-code	▪ Folder that contains the trained model ▪ Folder for notebooks ▪ Folder for feature engineering ▪ Folder for ML model engineering
Code	▪ Deployment process ▪ How to run locally	▪ Folder for bash/shell scripts ▪ Folder for tests ▪ Folder for deployment files (e. g. Docker files)

```
├── .cloud              # for storing cloud configuration files and templates (e.g. ARM, Terraform, etc)
├── .github
│   ├── ISSUE_TEMPLATE
│   │   ├── Ask.md
│   │   ├── Data.Aquisition.md
│   │   ├── Data.Create.md
│   │   ├── Experiment.md
│   │   ├── Explore.md
│   │   └── Model.md
│   ├── labels.yaml
│   └── workflows
├── .gitignore
├── README.md
├── code
│   ├── datasets        # code for creating or getting datasets
│   ├── deployment      # code for deploying models
│   ├── features        # code for creating features
│   └── models          # code for building and training models
├── data                # directory is for consistent data placement. contents are gitignored by default.
│   ├── README.md
│   ├── interim         # storing intermediate results (mostly for debugging)
│   ├── processed       # storing transformed data used for reporting, modeling, etc
│   └── raw             # storing raw data to use as inputs to rest of pipeline
├── docs
│   ├── code            # documenting everything in the code directory (could be sphinx project for example)
│   ├── data            # documenting datasets, data profiles, behaviors, column definitions, etc
│   ├── media           # storing images, videos, etc, needed for docs.
│   ├── references      # for collecting and documenting external resources relevant to the project
│   └── solution_architecture.md    # describe and diagram solution design and architecture
├── environments
├── notebooks
├── pipelines           # for pipeline orchestrators i.e. AzureML Pipelines, Airflow, Luigi, etc.
├── setup.py            # if using python, for finding all the packages inside of code.
└── tests               # for testing your code, data, and outputs
    ├── data_validation
    └── unit
```

https://github.com/dslp/dslp-repo-template

Quality Assurance & Governance

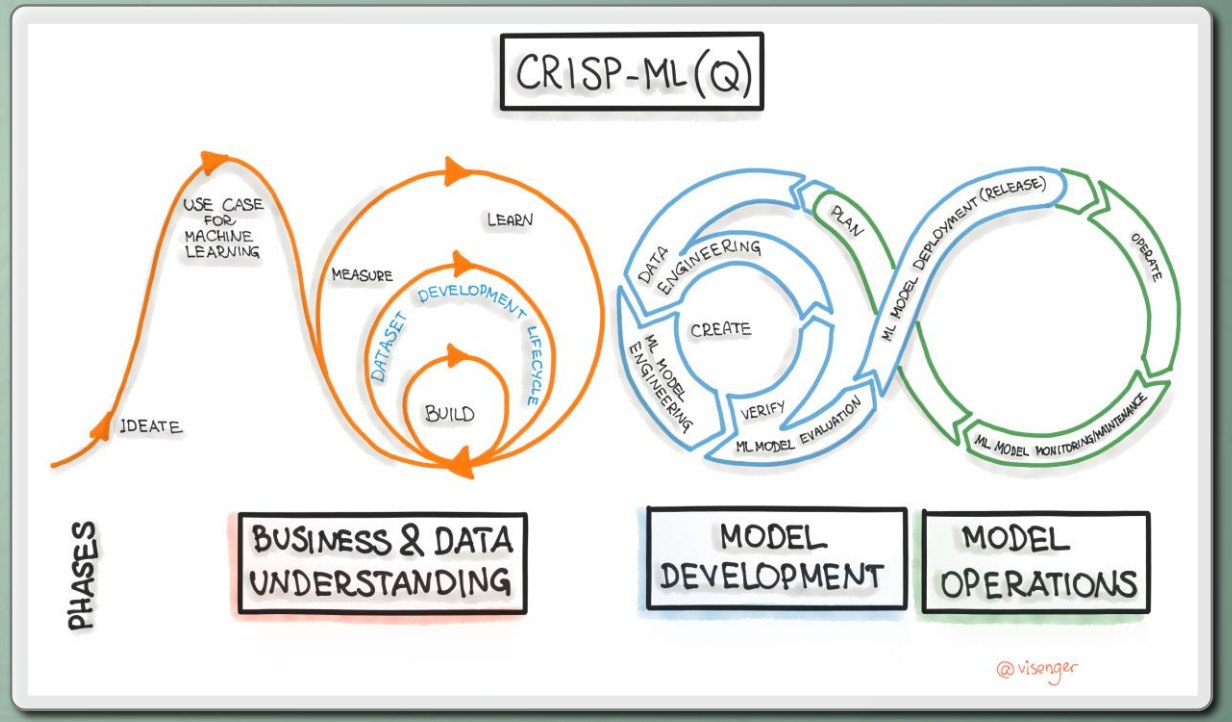
… involves the organization, summarization, and visualization of data. It provides simple summaries about the sample and the measures.



CRISP-ML(Q): The ML Lifecycle Process

... proposed standard to structure, enhance the organization of ML and data science projects.

Phase No.	Phase Name	Description
1	Business and Data Understanding	Setting context, understanding data sources.
2	Data Engineering (Data Preparation)	Processing and transforming raw data.
3	Machine Learning Model Engineering	Algorithm selection, training, and tuning.
4	Quality Assurance for ML Applications	Ensuring model robustness, performance, etc.
5	Deployment	Launching the model into a production or operational environment.
6	Monitoring and Maintenance	Continuous monitoring and updating the model as needed.



Model Governance

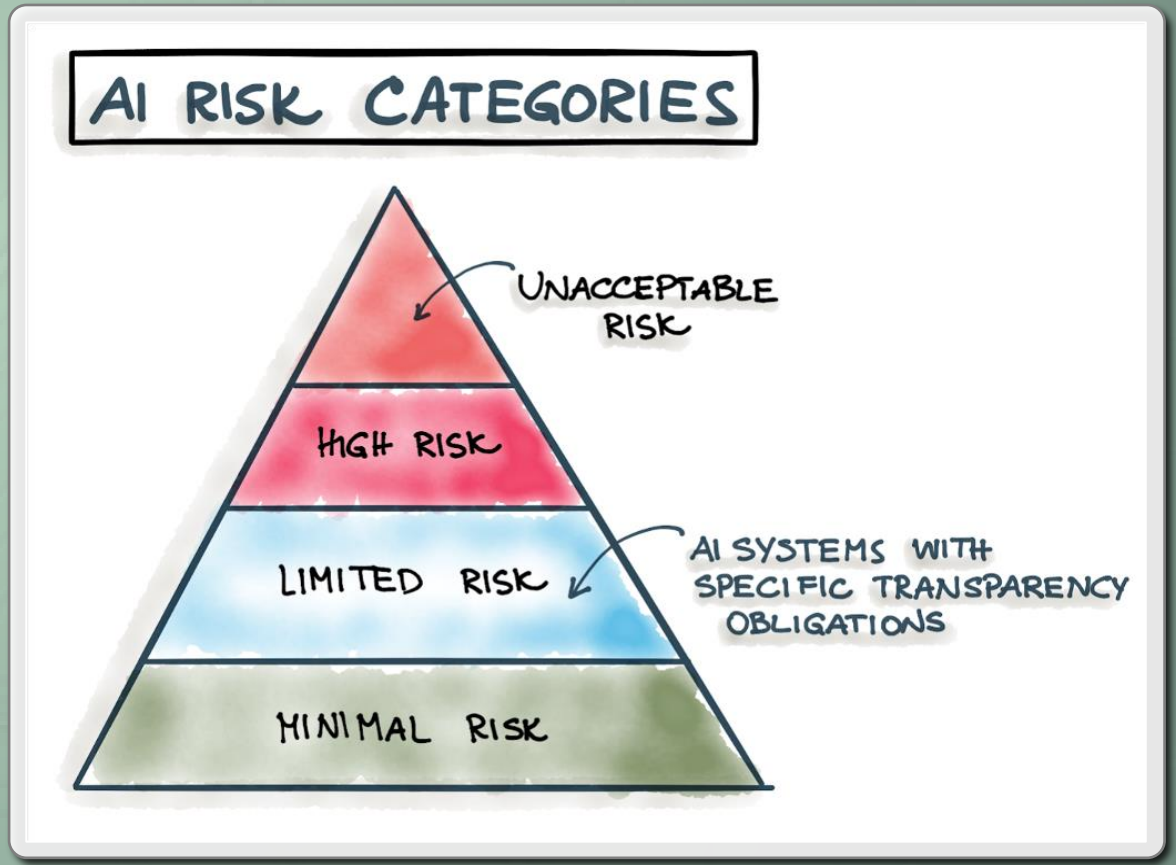
… becomes non-optional as companies navigate impending EU regulations, underscoring the need for rigorous AI system categorization and compliance strategies.

EU Regulation Overview

- April 2021 EU draft proposes the first legal framework for AI.
- Categorizes AI systems by risk levels, determining regulatory requirements.

AI Risk Categories

- Category 1: Unacceptable Risk
  High-risk AI (e.g., social scoring) banned to protect rights, security.
- Category 2: High Risk
  Stringent requirements for robustness, security, documentation, non-discrimination, and transparency. Systems include credit scoring and educational access.
- Category 3: Limited Risk
  Transparency obligations, such as informing users when interacting with AI (e.g., chatbots).
- Category 4: Minimal Risk
  Low or no regulation needed (e.g., spam filters).



Balancing Model Governance with MLOps

… is about navigating industry-specific regulations and to align with strategic deployment of ML models within company

Compliance is shaped by

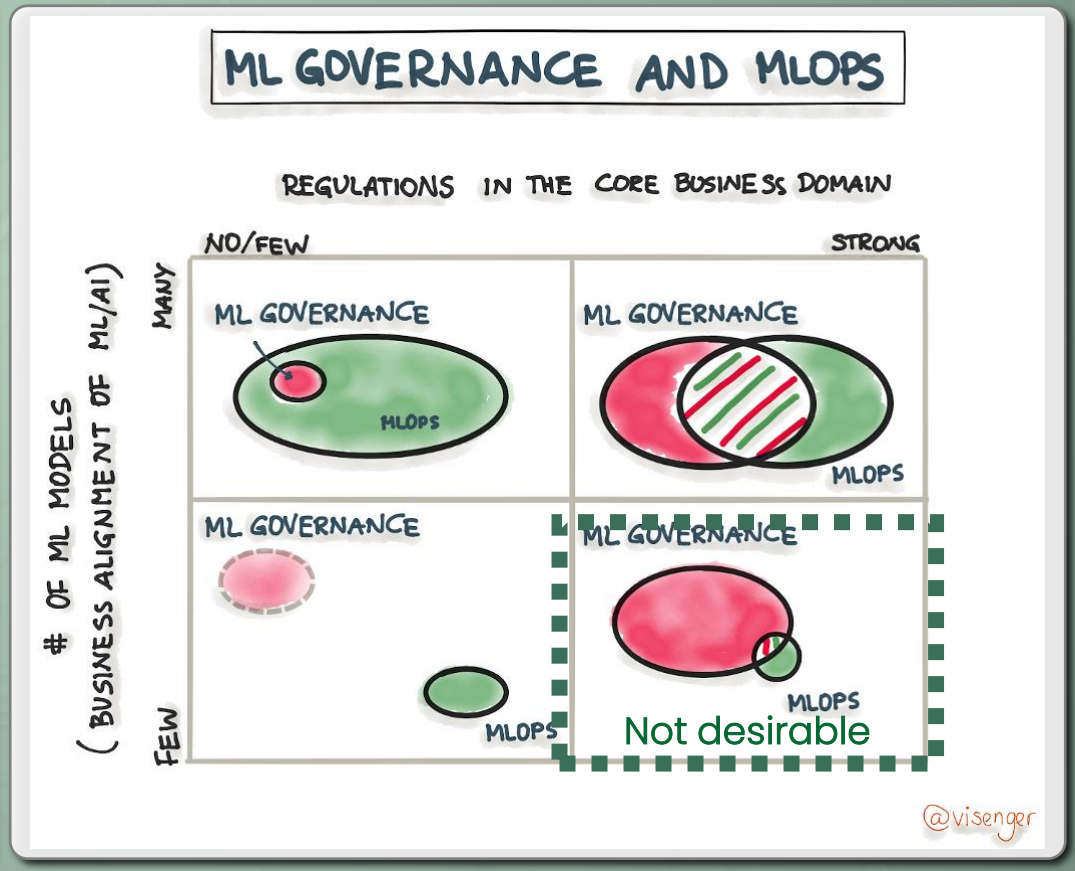
- business domain
- ML model risk category
- inherent business risks.

Model Proliferation

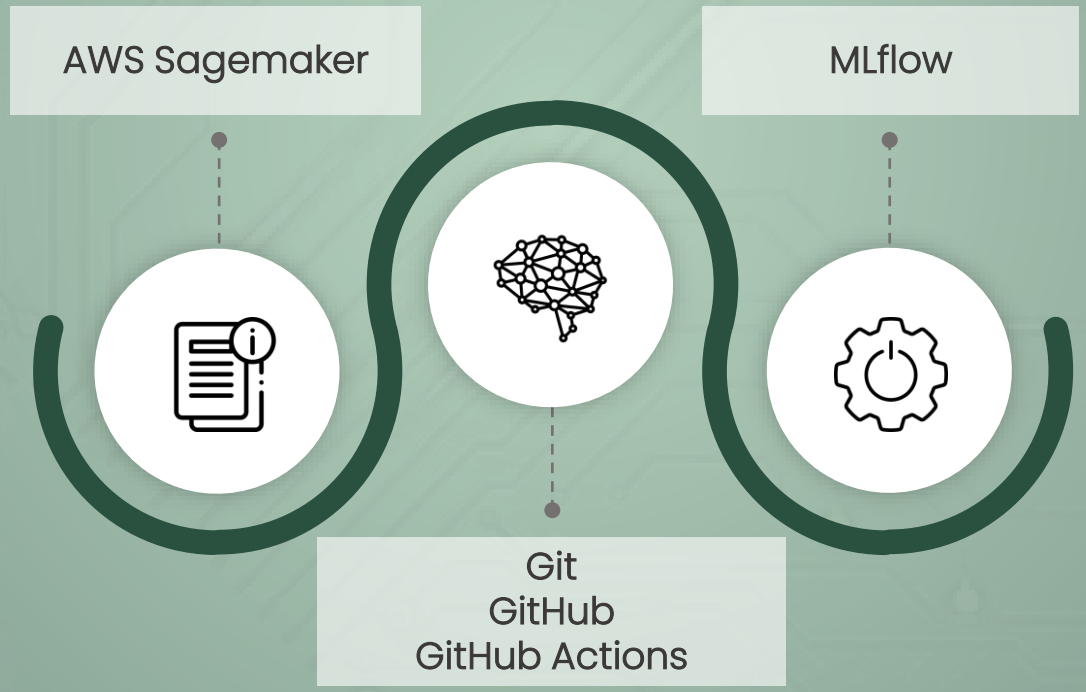
- Reflects ML's integration depth into business processes and company's maturity in ML implementation.

Model Count Implications

- High number of ML models indicates:
  extensive AI integration and advanced organizational ML capabilities.
- Lower count may signal:
  - either strategic choice / low demand
  - or developmental stage in ML adoption.



Technology & Tools



What is AWS SageMaker?

… fully managed service that provides developers and data scientists the ability to build, train, and deploy machine learning models quickly.

Key Features

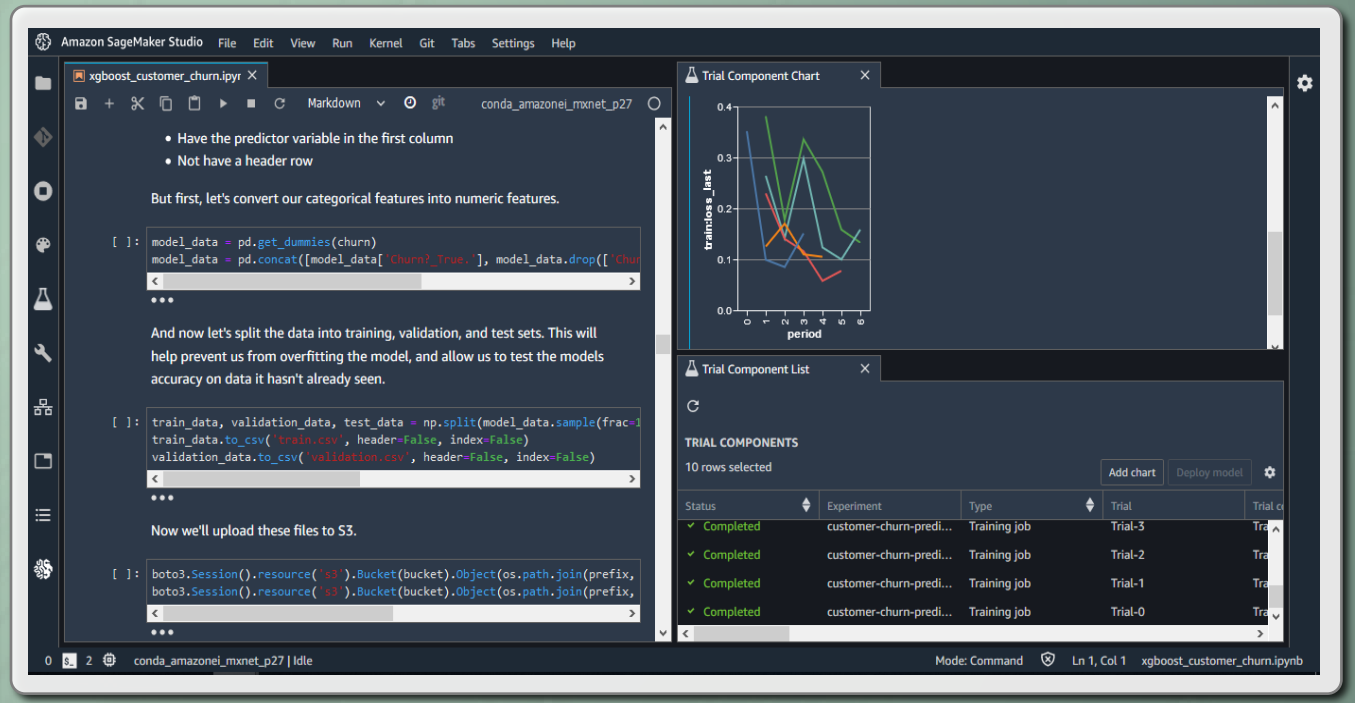
- Scalable Model Building:
  Utilize Jupyter notebooks or Studio with flexible instance types.
- Streamlined Model Training:
  One-click training with automatic model tuning.
- Effortless Deployment:
  Easy deployment of models for inference with auto-scaling.

Benefits

- Integrated ML Environment:
  Comprehensive tools for every step of the ML lifecycle.
- Reduced Complexity:
  Simplifies the process from concept to production.
- Cost-Effective:
  Pay-as-you-go pricing model, only for the resources used.

Use Cases

- ML for predictive analytics, personalization, and more.
- Startups innovating quickly by integrating ML into their products.



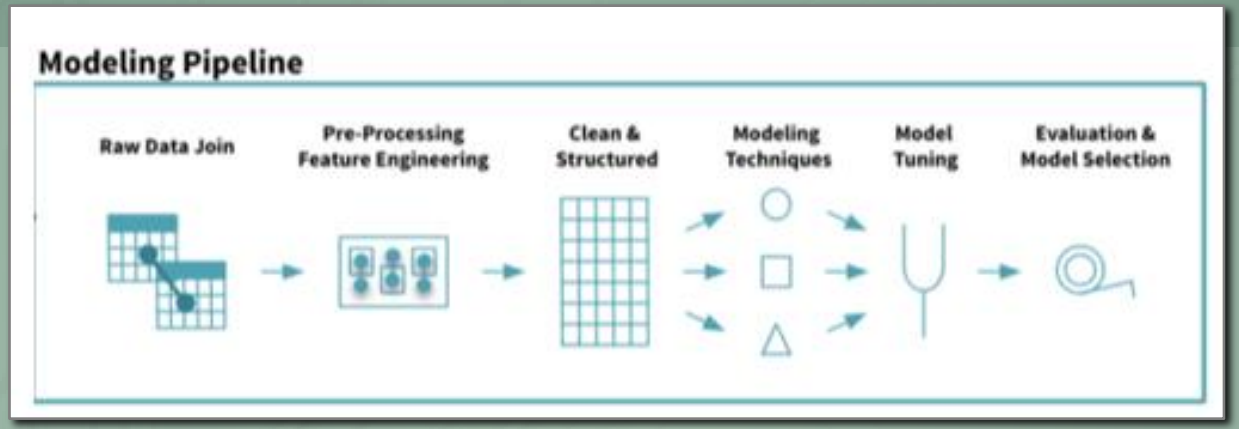
Git, GitHub, & GitHub Actions

... for mastering Version Control and Automation

	Git	GitHub	GitHub Actions
Definition	Distributed version control system for tracking changes in source code.	Cloud-based hosting service that utilizes Git for version control.	CI/CD platform integrated into GitHub to automate software workflows.
Key Features	Branching, merging, reverts, local repository management.	Issue tracking, project management, developer social networking, online repository hosting.	Automated workflows, customizable to project needs, triggered by GitHub events.
Benefits	Facilitates collaborative and individual development work.	Enhances collaboration, offers broad range of integration options.	Streamlines, automates tasks, from testing to deployment within GitHub ecosystem.
Use Cases	Source code management for solo and team projects.	Open-source, private development projects, community engagement, documentation hosting.	Continuous Integration, Continuous Deployment (CI/CD), testing automation, issue labeling.

MLflow: Managing the ML Lifecycle

Open-source platform designed to manage end-to-end machine learning lifecycle, encompassing creation, deployment and maintenance stages.



Benefits

- Unified Interface:
  Streamlines workflow for different ML tasks.
- Cross-platform:
  Compatible with various ML frameworks and languages.
- Community-Driven:
  Contributions from a wide range of users and companies.

Use Cases

- Ideal for teams seeking to streamline experimentation, reproducibility, and deployment of ML models.

mlflow

Tracking
Record and query experiments: code, data, config, results

Projects
Packaging format for reproducible runs on any platform

Models
General format for sending models to diverse deploy tools

Links

i.e. sources for self-learning

	Title	Link
What is ML Ops?	Machine Learning Operations	https://ml-ops.org/
	What is THE main reason most ML projects fail?	https://towardsdatascience.com/what-is-the-main-reason-most-ml-projects-fail-515d409a161f
	Practical MLOps: Operationalizing Machine Learning Models, Noah Gift	https://www.amazon.de/Practical-MLOps-Operationalizing-Machine-Learning/dp/1098103017

	Title	Link
Business Problems requiring ML	https://ml-ops.org/content/mlops-stack-canvas	https://ml-ops.org/content/mlops-stack-canvas
	Machine Learning Use Cases mit DDD und Design Canvas finden	https://www.youtube.com/watch?v=kFMUrxlbLeQ

Links

i.e. sources for self-learning

	Title	Link
MLOps Principles	MLOps: Continuous Delivery und Pipelines zur Automatisierung im maschinellen Lernen	https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning?hl=de#top_of_page
	Model versioning with Amazon Comprehend	https://docs.aws.amazon.com/comprehend/latest/dg/model-versioning.html

	Title	Link
ML Governance	The Framework for ML Governance	https://www.oreilly.com/library/view/the-framework-for/9781098100483/ch01.html
	CRISP-ML(Q). The ML Lifecycle Process.	https://ml-ops.org/content/crisp-ml
	Session - 6 | CRISP-ML(Q) - Data Understanding- Part 6 | Data Science Using Python	https://www.youtube.com/watch?v=mqWZ03JQHwg

Links

i.e. sources for self-learning

	Title	Link
Technology & Tools	Introducing MLflow: an Open Source Machine Learning Platform	https://www.databricks.com/blog/2018/06/05/introducing-mlflow-an-open-source-machine-learning-platform.html
	MLflow: A Primer	https://towardsdatascience.com/mlflow-a-primer-6dfe6be48353
	Tobias Sterbak: Introduction to MLOps with MLflow	https://www.youtube.com/watch?v=IUF4s9SXnd4
	Understanding MLOps: a Review of “Practical Deep Learning at Scale with MLFlow” by Yong Liu	https://cloud4scieng.org/2022/07/08/understanding-mlops-a-review-of-practical-deep-learning-at-scale-with-mlflow-by-yong-liu/
	Deploy ML models with FastAPI, Docker, and Heroku | Tutorial	https://www.youtube.com/watch?v=h5wLuVDr0oc
	Top Model Versioning Tools for Your ML Workflow	https://neptune.ai/
	The AI Developer Platform – Weights and Biases	https://wandb.ai/site/
	Amazon SageMaker Studio	https://aws.amazon.com/de/sagemaker/studio/
	Amazon SageMaker Studio: The First Fully Integrated Development Environment For Machine Learning	https://aws.amazon.com/de/blogs/aws/amazon-sagemaker-studio-the-first-fully-integrated-development-environment-for-machine-learning/
	Industrializing an ML platform with Amazon SageMaker Studio	https://towardsdatascience.com/industrializing-an-ml-platform-with-amazon-sagemaker-studio-91b597802afe
	MLOps In Practice – How To Run Your Machine Learning Models In Production At Enterprise Scale	https://www.youtube.com/watch?v=LQzTyV5suQo

About me

Dr. Harald Stein

- Data Scientist ~ 7 years experience
- Algotrader ~ 4 years experience
- Ph.D. in Economics, Game Theory

- LinkedIn: https://www.linkedin.com/in/harald-stein-phd-1648b51a
- ResearchGate: https://www.researchgate.net/profile/Harald-Stein