

# DADS7305: MLOPs

Northeastern University

Instructor: Ramin Mohammadi

September 7, 2025

These materials have been prepared and sourced for the course **MLOPs** at Northeastern University. Every effort has been made to provide proper citations and credit for all referenced works.

If you believe any material has been inadequately cited or requires correction, please contact me at:

`r.mohammadi@northeastern.edu`

*Thank you for your understanding and collaboration.*

# The Machine Learning Project Lifecycle

## Deployment Example

## Deployment Example

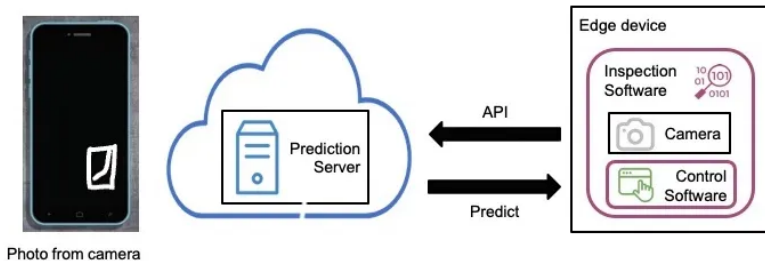


Figure: Deployment

## Visual inspection example



Figure: Visual inspection example

## ML In Production

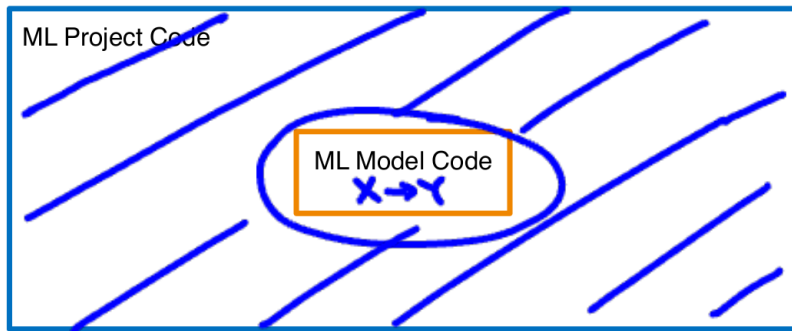
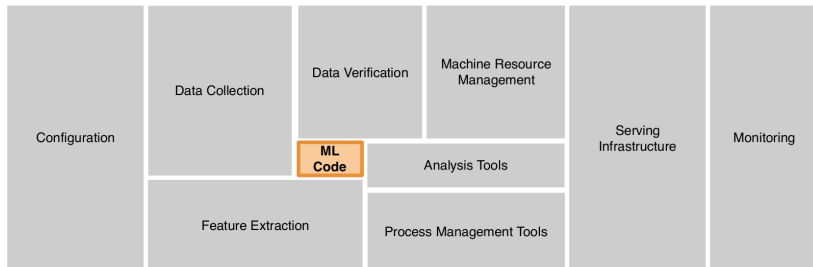


Figure: POC to Production Gap

## The requirements surrounding ML infrastructure



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

## Machine Learning Deployment Workflow

MLOps structures the ML system lifecycle into four main stages, each addressing critical engineering and operational challenges:

- ▶ **Data management** – Collection, preprocessing, and augmentation of data to ensure consistency and quality.
- ▶ **Model learning** – Model selection, training, and hyperparameter tuning within compute, interpretability, and resource constraints.
- ▶ **Model verification** – Validation against functional, regulatory, and business requirements through formal or test-based methods.
- ▶ **Model deployment** – Integration into production systems with support for monitoring, feedback, and continuous updates.

**Cross-cutting aspects** influence every stage of the pipeline:

- ▶ **Ethics** – Fairness, accountability, and decision transparency.
- ▶ **Law** – Adherence to regional regulations and compliance standards.
- ▶ **End users' trust** – Usability, explainability, and user involvement.
- ▶ **Security** – Protection against threats like model stealing, data poisoning, and inversion attacks.

This structure is not strictly linear-stages often run in parallel and inform each other through feedback loops. The following slides summarize common challenges at each step.



## Data Management

Deployment Stage	Deployment Step	Considerations, Issues and Concerns
Data management	Data collection	Data discovery
	Data preprocessing	Data dispersion, Data cleaning
	Data augmentation	Labeling of large volumes of data Access to experts, Lack of high-variance data
	Data analysis	Data profiling

## Model Learning

Deployment Stage	Deployment Step	Considerations, Issues and Concerns
Model learning	Model selection	Model complexity, Resource-constrained environments, Interpretability of the model
	Training	Computational cost, Environmental impact, Privacy-aware training
	Hyper-parameter selection	Resource-heavy techniques, Unknown search space, Hardware-aware optimization

## Model Verification

Deployment Stage	Deployment Step	Considerations, Issues and Concerns
Model verification	Requirement encoding	Performance metrics, Business driven metrics
	Formal verification	Regulatory frameworks
	Test-based verification	Simulation-based testing, Data validation routines, Edge case testing

## Model Deployment

Deployment Stage	Deployment Step	Considerations, Issues and Concerns
Model deployment	Integration	Operational support, Reuse of code and models, Software engineering anti-patterns, Mixed team dynamics
	Monitoring	Feedback loops, Outlier detection, Custom design tooling
	Updating	Concept drift, Continuous delivery

## Cross-cutting Aspects

Deployment Stage	Deployment Step	Considerations, Issues and Concerns
Cross-cutting aspects	Ethics	Aggravation of biases, Fairness and accountability, Authorship, Decision making
	Law	Country-level regulations, Abiding by existing legislation, Focus on technical solution only
	End users' trust	Involvement of end users, User experience, Explainability score
	Security	Data poisoning, Model stealing, Model inversion

## What Can Go Wrong in ML Deployment? – Part 1

### **Data Issues**

- ▶ Missing or incomplete data
- ▶ Data drift over time
- ▶ Labeling errors or inconsistencies
- ▶ Leakage between train/test splits

### **Model Issues**

- ▶ Overfitting or underfitting
- ▶ Poor generalization to unseen data
- ▶ Lack of interpretability
- ▶ Biases embedded in the model

### **Training Issues**

- ▶ Insufficient computational resources
- ▶ Hyperparameter misconfiguration
- ▶ Unstable training dynamics
- ▶ Incomplete convergence or premature stopping

## What Can Go Wrong in ML Deployment? – Part 2

### **Deployment Issues**

- ▶ Mismatch between training and production environments
- ▶ Inference latency or memory bottlenecks
- ▶ Integration failures with existing systems
- ▶ Lack of rollback or CI/CD support

### **Monitoring and Maintenance Issues**

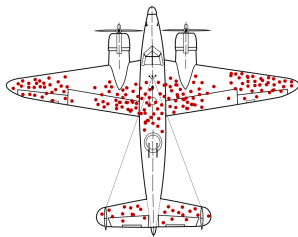
- ▶ No alerting for model degradation
- ▶ Failure to detect concept drift
- ▶ Lack of A/B testing or shadow deployment
- ▶ Manual, error-prone update process

### **Cross-cutting Issues**

- ▶ Security vulnerabilities (e.g., model stealing)
- ▶ Legal non-compliance (e.g., data privacy)
- ▶ Lack of user trust and transparency
- ▶ Ethical concerns (e.g., fairness violations)

## WWII - Survivorship Bias

During WWII, a study of bullet hole patterns on returning aircraft led to a counterintuitive insight about survivorship bias. The military planned to reinforce areas with the most bullet holes (fuselage, wings, tail).



**What could go wrong?**



## The Machine Learning Project Lifecycle

---

### Steps of an ML project

## The ML project lifecycle

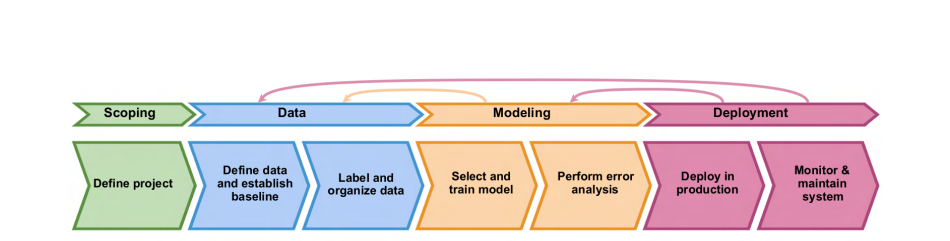


Figure: The ML project lifecycle

- ▶ Scoping
- ▶ Data
- ▶ Modeling
- ▶ Deployment

## The Machine Learning Project Lifecycle

---

### Case study: speech recognition

## Speech recognition: Scoping stage

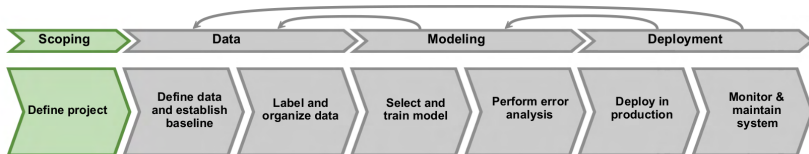


Figure: The ML project lifecycle

- ▶ Decide to work on speech recognition for voice search.
- ▶ Decide on key metrics:
  - ▶ Accuracy, latency, throughput
- ▶ Estimate resources and timeline

## Speech recognition: Data stage

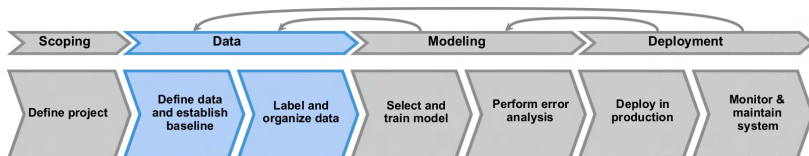


Figure: The ML project lifecycle

### Define data:

- ▶ Is the data labeled consistently?
- ▶ How much silence before/after each clip?
- ▶ How to perform volume normalization?

### Examples:

- ▶ “Um, today’s weather”
- ▶ “Um. . . today’s weather”
- ▶ “Today’s weather”

## Speech recognition: Modeling stage

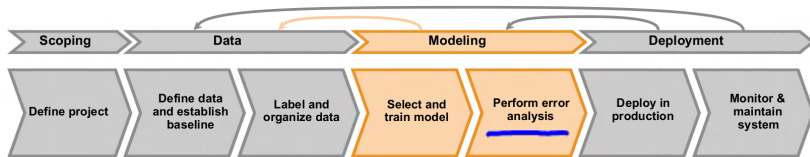
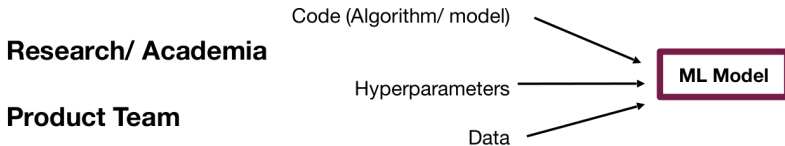


Figure: The ML project lifecycle



## Speech recognition: Deployment stage

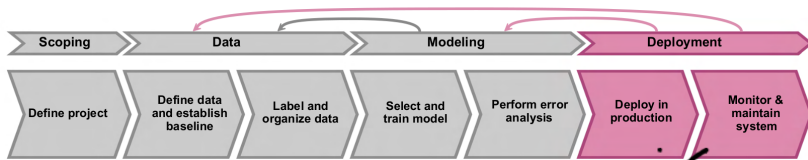
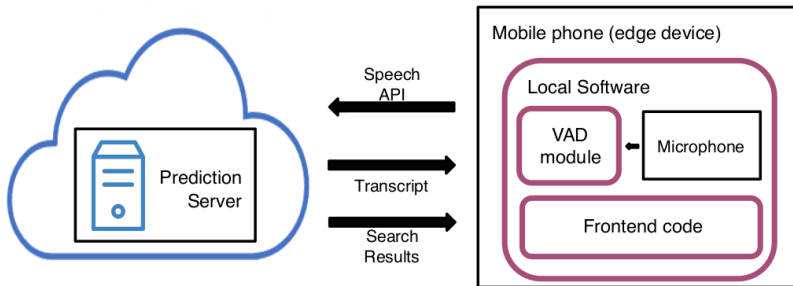


Figure: The ML project lifecycle



## Key requirements for an MLOps foundation

### AI-driven organizations are transforming industries:

- ▶ Leverage data and machine learning to tackle their hardest problems.
- ▶ According to McKinsey, companies that adopt AI across workflows by 2025 may see **+120% cash flow growth by 2030.**

### But operationalizing ML is not easy:

- ▶ ML systems incur significant technical debt if unmanaged.
- ▶ Combine traditional software issues *plus* ML-specific challenges:
  - ▶ Complex hardware/software dependencies
  - ▶ Data needs to be validated, not just code
  - ▶ Models degrade over time due to changing environments
  - ▶ Fail silently; harder to debug and monitor
- ▶ **Building a model is easy; managing its lifecycle is hard.**



## Bridging ML and IT: MLOps Meets DevOps

### **DevOps: Proven practice for large-scale software systems**

- ▶ Evolved over decades of software engineering experience.
- ▶ Benefits:
  - ▶ Shorter development cycles
  - ▶ Faster deployment velocity
  - ▶ Reliable, high-quality releases

### **MLOps: Extending DevOps to Machine Learning**

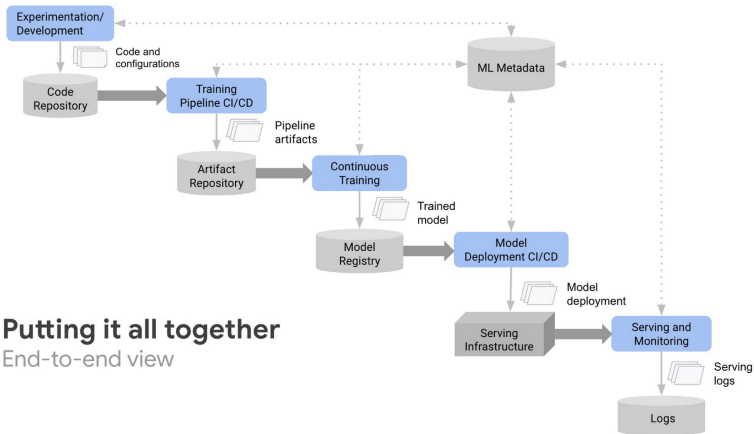
- ▶ MLOps = ML system development (Dev) + ML system operation (Ops)
- ▶ Inspired by DevOps, but adapted for the ML lifecycle.
- ▶ Challenges:
  - ▶ Continuous Integration / Delivery (CI/CD) is harder in ML
  - ▶ ML systems involve data, models, code, and environments

## CI/CD in ML Systems: Beyond Traditional Software

**ML systems redefine CI/CD and introduce new requirements:**

- ▶ **Continuous Integration (CI):**
  - ▶ Validate not just code and components
  - ▶ Also validate data, data schemas, and trained models
- ▶ **Continuous Delivery (CD):**
  - ▶ Not just deploying software
  - ▶ Deliver end-to-end ML pipelines that deploy model services
- ▶ **Continuous Training (CT):**
  - ▶ Unique to ML systems
  - ▶ Automate retraining of models for testing and deployment
- ▶ **Continuous Monitoring (CM):**
  - ▶ Monitor more than just system errors
  - ▶ Track live inference data and model performance degradation

## End-to-End ML Pipeline



# The Machine Learning Project Lifecycle

---

## LLMOps

## LLMOps vs MLOps

### LLMOps is a specialized extension of MLOps:

- ▶ **MLOps:** General practices for managing ML models ; includes data pipelines, training, deployment, monitoring, and CI/CD.
- ▶ **LLMOps:** Focused on large language models (LLMs), addressing unique challenges:
  - ▶ Extremely large model sizes
  - ▶ Complex multi-stage training
  - ▶ High compute and storage demands

### Core Components of LLMOps:

- ▶ **Data Collection & Preparation:** Massive, diverse, and clean datasets required for LLM pretraining.
- ▶ **Model Development:** Often involves unsupervised, supervised, and reinforcement learning techniques.
- ▶ **Model Deployment:** Requires scalable infrastructure and specialized serving configurations (e.g., GPUs/TPUs).
- ▶ **Model Management:** Ongoing monitoring, evaluation, retraining, and securing of models in production.

## What is LLMOps and Why is it Needed?

- ▶ **LLMOps** = Practices, tools, and workflows for managing the lifecycle of Large Language Models (LLMs) from development to deployment.
- ▶ Sub-discipline of MLOps tailored to the unique challenges of LLMs.
- ▶ Traditional ML: produces a single prediction/score.
- ▶ LLMs: generate complex language outputs ; harder to evaluate and control.
- ▶ Challenges: unpredictable behavior, hallucinations, inappropriate responses.
- ▶ Goal: Ensure LLM-powered applications are reliable and scalable in production.

## Why LLMOps Matters

- ▶ LLMs are larger, more complex, and often pre-trained by third parties.
- ▶ Without LLMOps:
  - ▶ Unpredictable outputs
  - ▶ High computational costs
  - ▶ Ethical risks
- ▶ LLM performance judged by human feedback for factuality and helpfulness.
- ▶ Practices include:
  - ▶ Human-in-the-loop evaluations
  - ▶ Prompt management
  - ▶ Safety checks

## LLMOps vs Traditional MLOps (Part 1)

### Shared Aspects:

- ▶ Dataset management
- ▶ Version control
- ▶ CI/CD deployment pipelines

### Key Differences:

- ▶ **Evaluation:** Harder to quantify, often requires human evaluation.
- ▶ Custom evaluation sets, RLHF, monitoring hallucinations/toxicity.

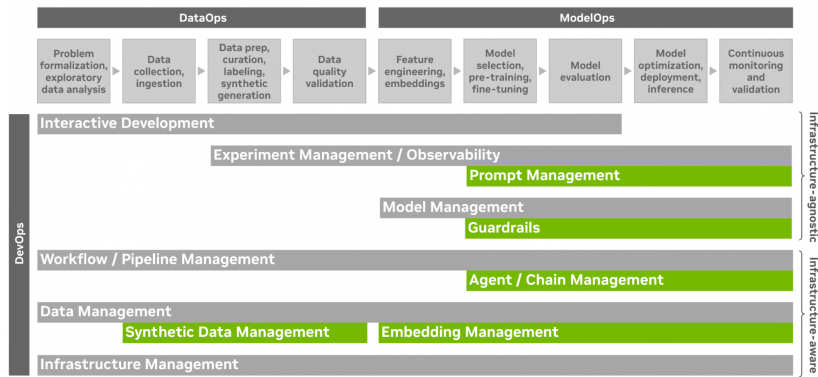


## LLMOps vs Traditional MLOps (Part 2)

- ▶ **Model Adaptation:** Start from pre-trained models (GPT, LLaMA).
- ▶ Adaptation techniques:
  - ▶ Prompt engineering
  - ▶ Fine-tuning on domain data
  - ▶ Retrieval Augmented Generation (RAG)
- ▶ **Infrastructure:** LLMs require GPUs, optimization, cost control.
- ▶ Techniques:
  - ▶ Request batching
  - ▶ Response caching
  - ▶ Routing to smaller models

## LLMOps vs Traditional MLOps (Part 3)

- ▶ **Monitoring Focus:**
  - ▶ Traditional ML: Accuracy, data drift.
  - ▶ LLMOps: Output content, hallucinations, bias, prompt injection.
- ▶ Summary: “MLOps for LLMs” + workflows for prompt management, feedback, ethical guardrails.



**Figure:** An end-2-end machine learning lifecycle showcasing core MLOps (gray) and GenAIOps capabilities (green)

## Core Components of the LLMOps Lifecycle

1. Data Collection and Preparation
2. Prompt Engineering and Embeddings
3. Fine-Tuning and Adaptation
4. LLM Chains and Agents
5. Evaluation and Testing
6. Deployment, Serving, and Monitoring
7. Governance and Security

## 1. Data Collection and Preparation

- ▶ Collect large text corpora from multiple sources.
- ▶ Clean, preprocess, normalize formats.
- ▶ Remove duplicates, irrelevant content.
- ▶ Ensure diversity and domain-representative coverage.

## 2. Prompt Engineering and Embeddings

- ▶ Iteratively design prompts to elicit desired outputs.
- ▶ Use instruction templates or few-shot examples.
- ▶ Manage embeddings for semantic search or RAG.
- ▶ Store and update embeddings in vector databases.

### 3. Fine-Tuning and Adaptation

- ▶ Fine-tune on domain-specific data.
- ▶ Use parameter-efficient methods (e.g., LoRA).
- ▶ Tailor vocabulary, tone, and style to application needs.
- ▶ Improves relevance, accuracy, and user satisfaction.

## 4. LLM Chains and Agents

- ▶ Go beyond single prompt→response.
- ▶ Chain multiple calls and tools.
- ▶ Agents can plan, retrieve info, and execute multi-step solutions.
- ▶ Use orchestration frameworks (e.g., LangChain).



## 5. Evaluation and Testing

- ▶ Continuous evaluation with structured rubrics.
- ▶ Human review and automated checks.
- ▶ Compare model versions side-by-side.
- ▶ Detect toxic, biased, or factually incorrect outputs.

## 6. Deployment, Serving, and Monitoring

- ▶ Deploy as API, containerized service, or backend integration.
- ▶ Monitor latency, throughput, cost per request.
- ▶ Track output quality, drift, and anomalies.
- ▶ Feed failures back into improvement cycles.

## 7. Governance and Security

- ▶ Control access to models and data.
- ▶ Enforce compliance with privacy and safety standards.
- ▶ Manage sensitive data flows and logs.
- ▶ Add content moderation and safety filters.

## Benefits of LLMOps

**LLMOps enables organizations to manage LLMs more effectively across the lifecycle:**

- ▶ **Performance:**

- ▶ Identify bottlenecks and optimize model parameters
- ▶ Improve accuracy, latency, and user experience

- ▶ **Scalability:**

- ▶ Flexible framework to adapt to increasing workloads and evolving requirements

- ▶ **Risk Reduction:**

- ▶ Proactive monitoring, disaster recovery, and security audits
- ▶ Reduces risk of outages, breaches, and disruptions

- ▶ **Efficiency:**

- ▶ Automates data prep, training, deployment, and monitoring
- ▶ Minimizes manual work and accelerates model iteration cycles

## Best Practices for LLMOps: Data Management

**Effective LLMOps begins with robust data practices:**

- ▶ **Use High-Quality Data**
  - ▶ Ensure data is clean, accurate, and relevant to the target domain.
  - ▶ High-quality input data leads to more performant and generalizable LLMs.
- ▶ **Manage Data Efficiently**
  - ▶ Handle large-scale data generated during training and inference.
  - ▶ Employ techniques such as compression, partitioning, and streaming for scalable data handling.
- ▶ **Establish Data Governance**
  - ▶ Define clear policies for data access, usage, and compliance.
  - ▶ Promote responsible data stewardship across the LLM lifecycle.

## Best Practices for LLMOps: Model Training

**Optimizing model training is essential for LLM performance:**

- ▶ **Choose the Right Training Algorithm**
  - ▶ Select algorithms that best fit your model architecture and task (e.g., pretraining, fine-tuning).
  - ▶ Evaluate based on scalability, convergence speed, and resource constraints.
- ▶ **Optimize Training Parameters**
  - ▶ Perform hyperparameter tuning (e.g., learning rate, batch size, weight decay).
  - ▶ Use automated tools (e.g., Optuna, Ray Tune) for large-scale optimization.
- ▶ **Monitor Training Progress**
  - ▶ Track metrics such as training/validation loss, accuracy, and perplexity.
  - ▶ Use dashboards to detect early signs of overfitting or divergence.

## Best Practices for LLMOps: Model Deployment

**Deploying LLMs reliably and securely is critical to production success:**

- ▶ **Choose the Right Deployment Strategy**
  - ▶ Options include cloud services, on-premises infrastructure, or edge devices.
  - ▶ Select based on latency, compliance, cost, and scalability needs.
- ▶ **Optimize Deployment Performance**
  - ▶ Monitor inference latency, throughput, and resource utilization.
  - ▶ Apply optimizations like quantization, caching, or model distillation.
- ▶ **Ensure Security**
  - ▶ Enforce access control, data encryption, and secure APIs.
  - ▶ Conduct regular audits to detect vulnerabilities and maintain compliance.

## Best Practices for LLMOps: Monitoring

**Monitoring ensures reliability, performance, and continuous improvement:**

- ▶ **Establish Monitoring Metrics**

- ▶ Define KPIs such as accuracy, latency, throughput, and memory usage.
- ▶ Tailor metrics to align with business and model-specific goals.

- ▶ **Implement Real-Time Monitoring**

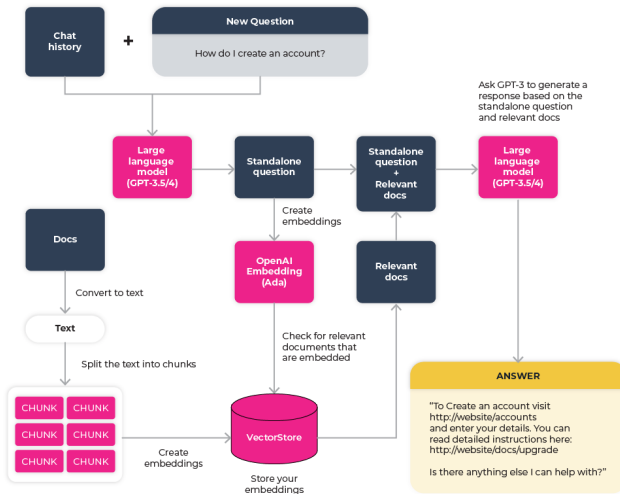
- ▶ Detect and respond to anomalies or drift during inference.
- ▶ Use observability stacks with alerting and logging tools (e.g., Prometheus, Grafana).

- ▶ **Analyze Monitoring Data**

- ▶ Continuously evaluate trends and usage patterns.
- ▶ Use insights to guide retraining, scaling, and process improvements.



## LLMOps Example:

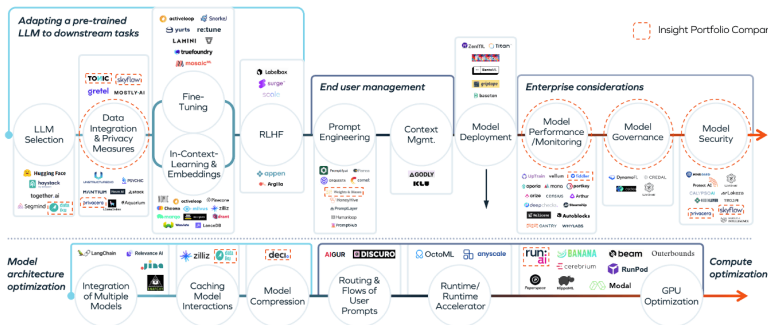
PRODUCT DOCUMENTATION  
CHATBOT ARCHITECTURE

## LLMOps Steps:

LLMOps adapts the MLOps tech stack for generative AI use cases

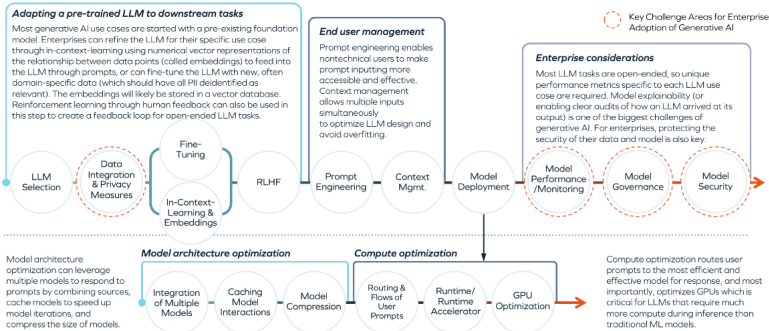
INSIGHT  
PARTNERS

Insight Portfolio Company



## LLMOps Steps:

## LLMOps adapts the MLops tech stack for generative AI use cases

INSIGHT  
PARTNERS

# The Machine Learning Project Lifecycle

---

## Course outline

## Course outline

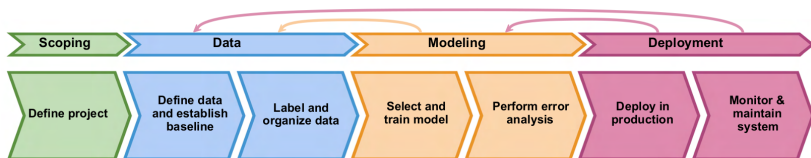


Figure: The ML project lifecycle

1. Scoping
2. Data
3. Modeling
4. Deployment

*MLOps (Machine Learning Operations) is an emerging discipline, and comprises a set of tools and principles to support progress through the ML project lifecycle.*

## Labs for This Week

### Objective

Learn how to setup GitHub Action/workflows.

### Lab Activities:

- ▶ Lab 1: [Github] ; [Github Workflow]

**Submission Deadline:** [Before the next class]

- ▶ Assignment 1: [Github] ; [Github Workflow]

## Reading Materials

### This Week's Theme

Topic focus: [Hidden Technical Debt in Machine Learning Systems]

### Required Readings:

- ▶ [Hidden Technical Debt in Machine Learning Systems]

*Be prepared to discuss highlights and open questions in class.*



DeepLearning.AI