Module 1: A Gentle Introduction to AI and ML

LLMOps: Foundations, Deployment, and Responsible Operations of Large Language Models

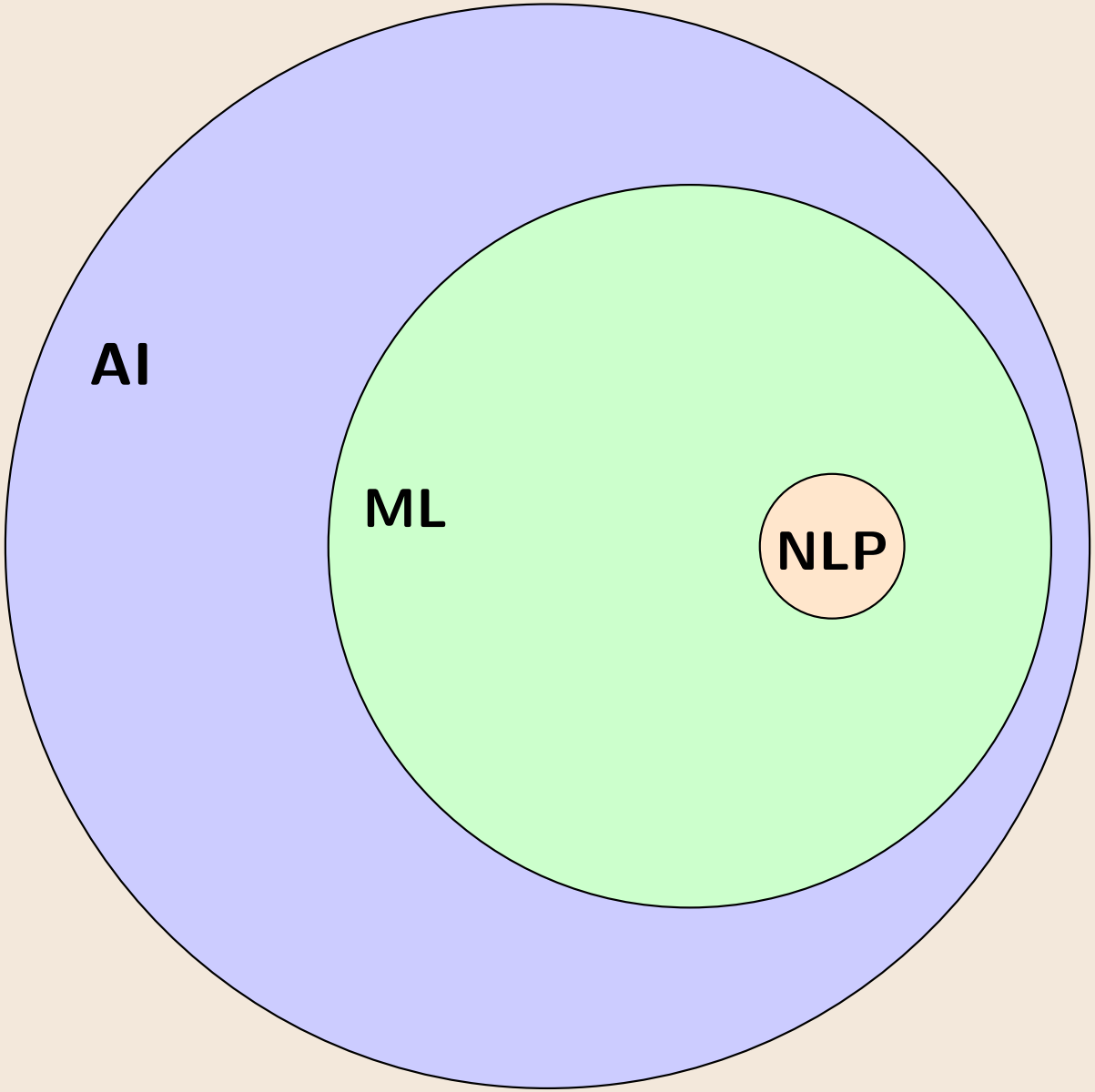
Module 1: Learning Objectives

- Define Artificial Intelligence (AI) and its core capabilities, including perception, reasoning, learning, language processing, and decision-making.
- Trace the historical evolution of AI, from symbolic systems to modern deep learning and foundation models.
- Differentiate AI paradigms (symbolic, statistical, neuro-symbolic) and types (narrow, general, superintelligence).
- Understand the socio-technical implications of AI, such as bias, privacy, environmental impact, and economic effects.
- Explore AI applications across industries like healthcare, finance, and education, and identify future challenges including explainability and governance.
- Grasp machine learning fundamentals, including paradigms (supervised, unsupervised, reinforcement), workflows, tools, and emerging trends like MLOps and responsible AI.

AI Domains

- **AI** (Artificial Intelligence): encompasses all intelligent systems (largest scope).
- **ML** (Machine Learning): a core subset of AI focusing on data-driven learning algorithms.
- **NLP** (Natural Language Processing): a further subset of ML specializing in human language.
- **LLMs** (Large Language Models): the most specialized subset—massive transformer models for language tasks.

AI, ML, and NLP



Overview of Artificial Intelligence

- Define AI and its core capabilities
- Trace AI's historical evolution
- Contrast AI paradigms and types
- Explore socio-technical considerations
- Survey modern applications and future directions

What Is Artificial Intelligence?

- AI: multidisciplinary field building systems that mimic human intelligence—learning, reasoning, perception, language, decision-making
- Core capabilities:
  - **Perception**: sensing & interpreting data
  - **Reasoning**: drawing logical inferences
  - **Learning**: improving from experience
  - **Language**: processing & generating human language
  - **Decision-making**: selecting actions by goals/context

Types of AI by Capability

- **ANI (Artificial Narrow Intelligence)** Systems specialized to perform a single task or a narrow range of tasks (e.g., image classifiers, chess engines, voice assistants).
- **AGI (Artificial General Intelligence)** Hypothetical systems with the ability to understand, learn, and apply knowledge across a wide variety of domains at human-level proficiency.
- **ASI (Artificial Superintelligence)** Speculative future systems that would outperform the best human experts in virtually every cognitive task, including creativity, social skills, and scientific reasoning.

Paradigms of AI: Symbolic AI

Symbolic AI (GOFAI)

- **Rule-based** systems using predicate logic
- **Transparent** reasoning chains
- **Inflexible**: struggles with unanticipated inputs or edge cases

Note: GOFAI stands for *Good Old-Fashioned Artificial Intelligence*.

Paradigms of AI: Statistical AI

Statistical AI

- **Data-driven** learning (ML, neural networks)
- **Adaptable** to large, diverse datasets
- **Opaque** (“black-box”) decision processes

Paradigms of AI: Neuro-Symbolic Hybrids

Neuro-Symbolic Hybrids

Integrate explicit symbolic reasoning (rules, logic, knowledge graphs) with neural/statistical learning (embeddings, pattern discovery) to combine interpretability, domain knowledge, and data-driven flexibility.

Historical Waves of AI

- **Origins (Antiquity–1950s)**
  - Aristotle's syllogistic logic
  - Al-Khwarizmi's algorithmic methods
- **Symbolic Era (1956–1980s)**
  - 1956 Dartmouth Conference: birth of AI
  - GOFAI systems (Logic Theorist, expert systems)
- **Statistical Era & AI Winters (1970s–1990s)**
  - Decline of early symbolic approaches ("AI winters")
  - Resurgence via probabilistic models & expert systems
- **Deep Learning & Foundation Models (2000s–present)**
  - Breakthroughs in backpropagation, CNNs, RNNs
  - Emergence of transformers & large-scale pretrained models

AI Timeline & Key Milestones

- **1956**: Dartmouth Conference
- **1965**: Logic Theorist & ELIZA demonstrate early reasoning & dialogue
- **1970s–80s**: AI winters; rise of expert systems (MYCIN, PROLOG)
- **1986**: Revitalization via backpropagation (Rumelhart et al.)
- **2012**: ImageNet breakthrough with deep CNNs (AlexNet)
- **2017**: "Attention Is All You Need" introduces transformers
- **2020–2023**: Foundation models (BERT, GPT-3 & GPT-4) & multimodal AI

Socio-Technical Implications

- **Bias & Fairness**: data reflects societal prejudices
- **Privacy & Governance**: data rights vs. model power
- **Environmental Impact**: energy cost of training
- **Labor & Economy**: automation, upskilling, policy

Applications Across Industries

- Healthcare: imaging, diagnostics, personalized medicine
- Finance: fraud detection, risk modeling, robo-advisors
- Education: adaptive learning, automated assessment
- Environment: climate modeling, resource optimization
- Creative: generative art, music, design prototypes
- Multimodal & foundation-model services

Future Directions & Challenges

- Explainability & Trust: interpretable models
- Common-Sense Reasoning: contextual understanding
- Path to AGI: unknown timeline & technical hurdles
- Interdisciplinary Integration: ethics, cognitive science
- Governance & Regulation: safe, equitable deployment

Wrap up - AI

- AI spans from symbolic rules to deep foundation models
- Paradigms differ in transparency, flexibility, scope
- Socio-technical issues demand ethical, policy responses
- Applications are vast; research frontiers remain open

Overview: Machine Learning Fundamentals

- Historical trajectory and key eras
- Mathematical & statistical foundations
- Taxonomy of learning paradigms
- End-to-end ML workflow
- Tools, technologies & roles
- Emerging trends and reflections

Machine Learning Fundamentals

Formalization and overview

What Is Machine Learning?

- Study of algorithms that improve automatically through experience
- Powers modern AI: automates processes, enhances decisions, generates insights, enables new products
- Requires integration of statistics, CS & engineering

Key Eras in Machine Learning

Era	Milestone	Key Contributors
1940–1959	Cybernetics, Hebbian learning, early artificial neurons	McCulloch & Pitts; Donald Hebb
1960–1979	Perceptrons, statistical classifiers, pattern recognition	Rosenblatt; Minsky & Papert
1980–1999	Backpropagation revival, statistical learning theory, expert systems	Rumelhart; LeCun; Vapnik; Quinlan
2000–2012	Kernel methods, ensemble learning, Big Data era	Schölkopf; Breiman; Friedman
2012–present	Deep learning boom, self-supervision, transformers	Hinton; Bengio; Radford; Vaswani

Mathematical & Statistical Foundations

- **Probability Theory**: modeling uncertainty, Bayes' theorem
- **Linear Algebra**: vectors, matrices, tensors, decompositions
- **Optimization & Calculus**: gradient descent, convex programming
- **Information Theory**: entropy, KL divergence for representation learning
- **Statistical Learning Theory**: generalization bounds, VC-dimension

Taxonomy of Learning Paradigms

Paradigm	Supervision	Use Cases
Supervised	Fully labeled data	Classification, regression
Unsupervised	No labels	Clustering, dimensionality reduction
Semi-supervised	Few labels + unlabeled	Web content classification
Reinforcement	Reward signals	Robotics, game playing
Self-supervised	Proxy tasks	Language modeling, representation pretraining
Federated	Decentralized data	Privacy-preserving mobile & edge AI

Other Machine Learning Paradigms

- **Meta-Learning / Few-Shot Learning** Models that "learn to learn," adapting quickly from few examples.
- **Active Learning** Algorithms that query an oracle (e.g., human annotator) for labels on the most informative samples.
- **Transfer Learning** Reusing pretrained models or representations on new but related tasks.
- **Online / Continual Learning** Incrementally updating models from streaming or non-stationary data without forgetting.
- **Multi-Task Learning** Jointly training on multiple related tasks to improve generalization via shared representations.
- **Ensemble Learning** Combining multiple models (e.g., bagging, boosting, stacking) to reduce variance and bias.
- **Graph-Based Learning** Techniques (e.g., GNNs) that operate on graph-structured data to capture relational inductive biases.

Supervised Learning

Formalization and overview

What Is Supervised Learning?

Core Idea

Supervised learning is the process of learning a mapping from inputs to outputs by leveraging labeled examples, where each training instance provides both the observed data and the corresponding ground-truth label.

- Learning from **experience**: the model adjusts its parameters based on many examples.
- Anchored by **ground truth**: each example (x_i, y_i) tells the model the correct answer.
- Tasks include classification (discrete y) and regression (continuous y).

Formalizing Supervised Learning

- Given dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where:
  - $x_i \in \mathcal{X}$ are input features (experience).
  - $y_i \in \mathcal{Y}$ are ground-truth labels.
- Goal: learn predictor $f_\theta : \mathcal{X} \to \mathcal{Y}$ minimizing loss on $\mathcal{D}$.
- Empirical risk minimization:

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(f_\theta(x_i), y_i\big).$$

- The more diverse and representative the experience–label pairs, the better the model generalizes.

Note: $\mathcal{L}$ is the loss function.

Loss Functions

Definition

A *loss function* $\mathcal{L}(f_\theta(x),\, y)$ quantifies the error of a single prediction $f_\theta(x)$ against its ground-truth label y.

- **Squared Error (Regression)** $\mathcal{L}(f, y) = (y - f)^2$ penalizes large deviations symmetrically.
- **Cross-Entropy (Binary Classification)** $\mathcal{L}(p, y) = -\big[y \log p + (1 - y) \log(1 - p)\big]$ encourages accurate probability estimates.
- **Hinge Loss (SVM)** $\mathcal{L}(f, y) = \max\big(0,\, 1 - yf\big), \quad y \in \{-1, +1\}$ enforces a margin of at least 1 for correct classifications.

Empirical Risk Minimization (ERM)

Framework

ERM aggregates individual losses over the dataset and finds parameters θ that minimize their average.

$$\hat{R}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\big(f_\theta(x_i),\, y_i\big), \quad \theta^* = \arg\min_\theta \hat{R}(\theta).$$

- $\mathcal{L}$ = per-example error; $\hat{R}$ = average error (risk).
- Training adjusts θ via gradient-based optimizers (SGD, Adam) on $\hat{R}$.
- Choice of $\mathcal{L}$ tailors the model to regression, classification, margin objectives, etc.

Regression Loss Functions

- **Squared Error (L2)** $\mathcal{L}(f, y) = (y - f)^2 \rightarrow$ penalizes large deviations, convex, closed-form solution for linear models.
- **Absolute Error (L1)** $\mathcal{L}(f, y) = |y - f| \rightarrow$ robust to outliers, nondifferentiable at zero (use subgradients).
- **Huber Loss**

$$\mathcal{L}_\delta(f, y) = \begin{cases} \frac{1}{2}(y - f)^2, & |y - f| \leq \delta, \\ \delta\left(|y - f| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

$\rightarrow$ combines L2 (small errors) and L1 (large errors), smooth transition.

Classification Loss Functions

- **Binary Cross-Entropy** $\mathcal{L}(p, y) = -\big[y \log p + (1-y)\log(1-p)\big] \rightarrow$ measures log-probability of true class, encourages confident correct predictions.
- **Multi-class Cross-Entropy** $\mathcal{L}(\mathbf{p}, k) = -\log p_k$ where $p_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$ from logits z.
- **Hinge Loss (SVM)** $\mathcal{L}(f, y) = \max(0,\, 1 - y\, f)$ for $y \in \{-1, +1\}$. $\rightarrow$ enforces margin: correct examples must lie beyond a unit-wide "safe zone."
- **Focal Loss** $\mathcal{L}(p, y) = -(1-p)^{\gamma}\big[y \log p + (1-y)\log(1-p)\big] \rightarrow$ down-weights well-classified examples to focus learning on hard cases.

Common Supervised Learning Methods

- **Linear Models** Linear Regression, Logistic Regression
- **Margin-Based** Support Vector Machines (SVM)
- **Tree-Based** Decision Trees, Random Forests (Bagging), Gradient Boosting (XGBoost, LightGBM)
- **Instance-Based** k-Nearest Neighbors (k-NN)
- **Probabilistic** Naive Bayes (Gaussian, Multinomial)
- **Neural Networks** Multi-layer Perceptrons (MLP), Convolutional & Recurrent architectures

Linear Models: Linear Regression

Objective

Fit a line $y = w^\top x + b$ by minimizing the sum of squared errors:

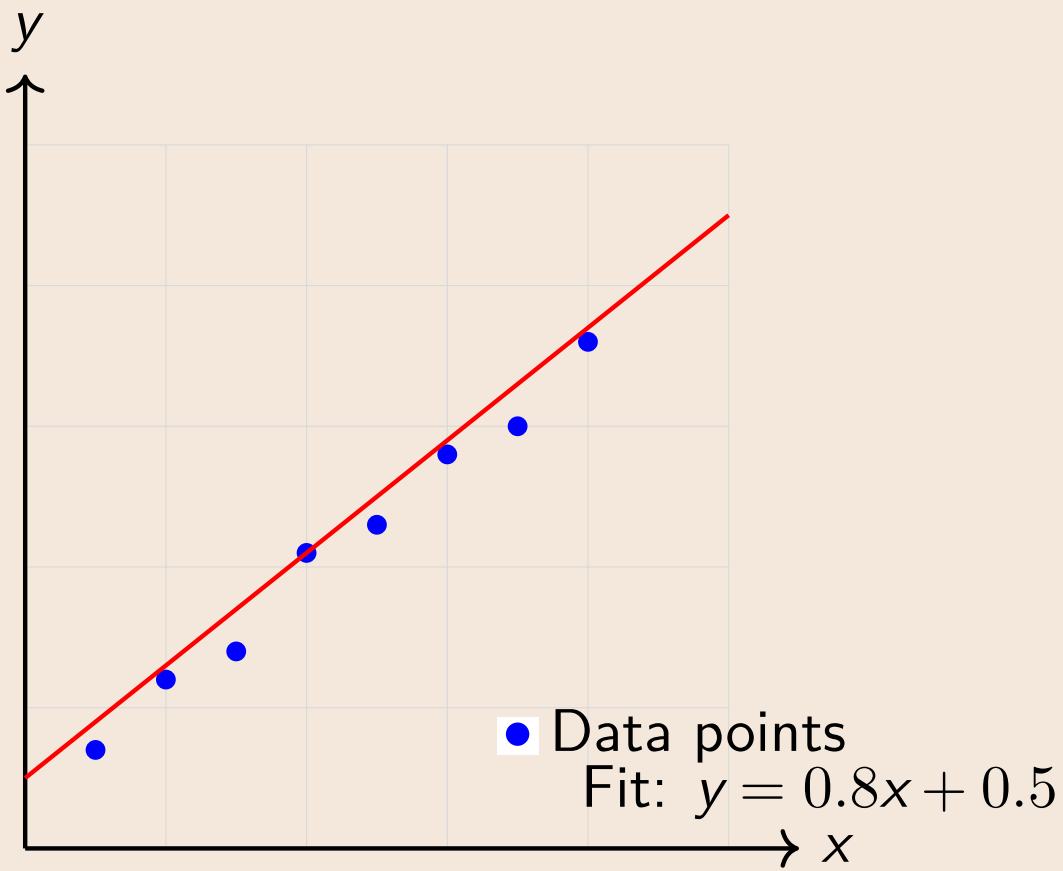
$$\min_{w,b} \sum_{i=1}^{N} (y_i - w^\top x_i - b)^2.$$

- **Closed-form solution:** $w^* = (X^\top X)^{-1} X^\top y$
- **Use cases:** forecasting, trend analysis, simple predictive modeling
- **Assumptions:** linear relationship, homoscedasticity, Gaussian errors

Note

A *closed-form solution* computes $w^* = (X^\top X)^{-1} X^\top y$ directly via matrix operations—no iterative optimization required.

Linear Regression Example



Linear Models: Logistic Regression

Objective

Model the probability of the positive class via the sigmoid function(e.g., $\sigma(z) = \frac{1}{1+e^{-z}}$),

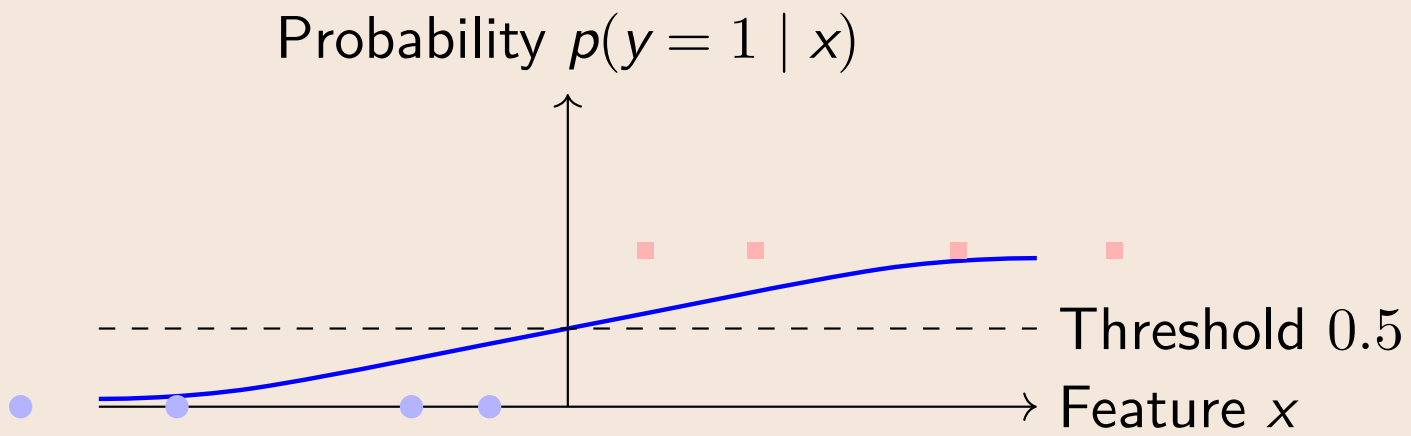
$$p(y = 1 \mid x) = \sigma(w^\top x + b),$$

and fit parameters by minimizing the log-loss:

$$\min_{w,b} \; -\sum_{i=1}^{N} \left[y_i \log p_i + (1 - y_i) \log(1 - p_i) \right].$$

- **Decision rule:** predict 1 if $p(y = 1 \mid x) \geq 0.5$
- **Use cases:** binary classification (e.g., spam detection).
- **Properties:** convex optimization, interpretable feature weights

Logistic Regression Illustration



- The blue curve shows the sigmoid $\sigma(w^\top x + b)$ mapping x to probability.
- We classify as positive when $p \geq 0.5$, i.e. $w^\top x + b \geq 0$ (vertical line).
- Blue circles (negative class) lie mostly left of the boundary; red squares (positive) lie right.

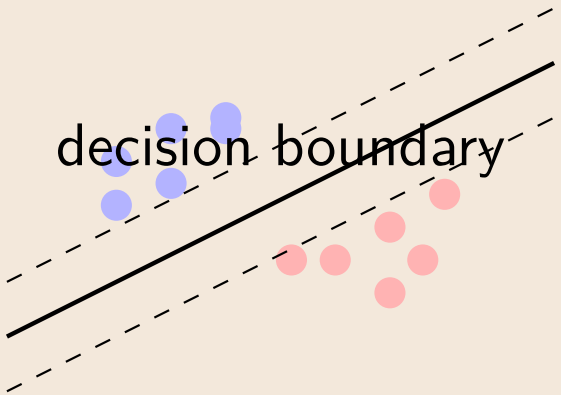
Margin-Based Method: SVM

Support Vector Machine

Find hyperplane $w^\top x + b = 0$ that maximizes margin:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t. } y_i(w^\top x_i + b) \geq 1.$$

Effective for high-dimensional, well-separated data.



Introduction to Decision Trees

What Is a Decision Tree?

A hierarchical, tree-structured model that makes predictions by recursively partitioning the feature space with simple tests.

- **Root node:** entire dataset; select a feature and threshold to split
- **Internal nodes:** tests of the form $x_j \leq t$
- **Branches:** outcomes of those tests (e.g., "yes" / "no")
- **Leaf nodes:** final predictions (class label or regression value)
- **Interpretability:** easy to visualize and explain decision paths
- **Limitations:** prone to overfitting; sensitive to small changes in data

Splitting Criterion: Entropy & Information Gain

Entropy

For a dataset S with class proportions p_c, entropy measures impurity:

$$H(S) = -\sum_{c} p_c \log_2 p_c, \quad p_c = \frac{|\{(x, y) \in S : y = c\}|}{|S|}.$$

$H(S) = 0$ when S is pure; maximum when classes are evenly mixed.

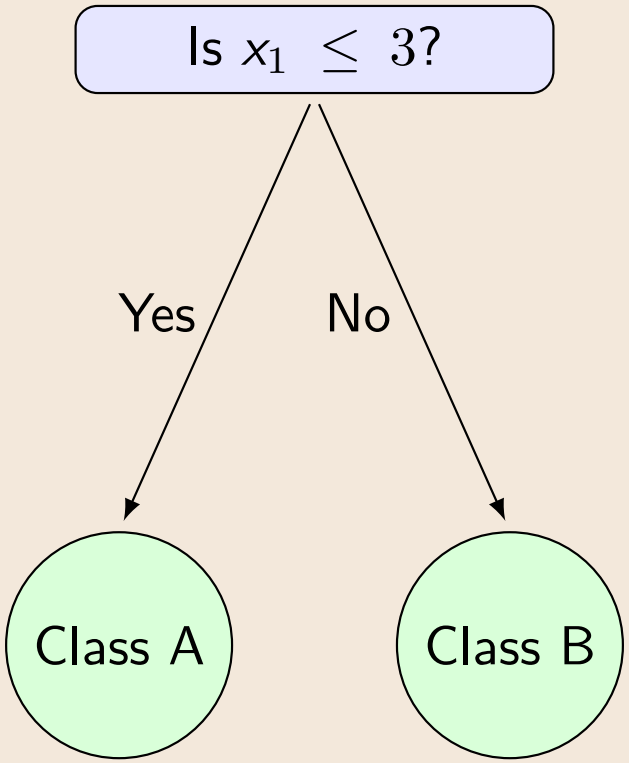
Information Gain

The reduction in entropy achieved by splitting S on feature A (threshold t):

$$\text{Gain}(S, A, t) = H(S) - \sum_{v \in \{\leq t, > t\}} \frac{|S_v|}{|S|} H(S_v),$$

where S_v is the subset of S satisfying the split condition $A(x)$ v t.

Decision Tree Structure



Instance-Based Methods

k-Nearest Neighbors (k-NN)

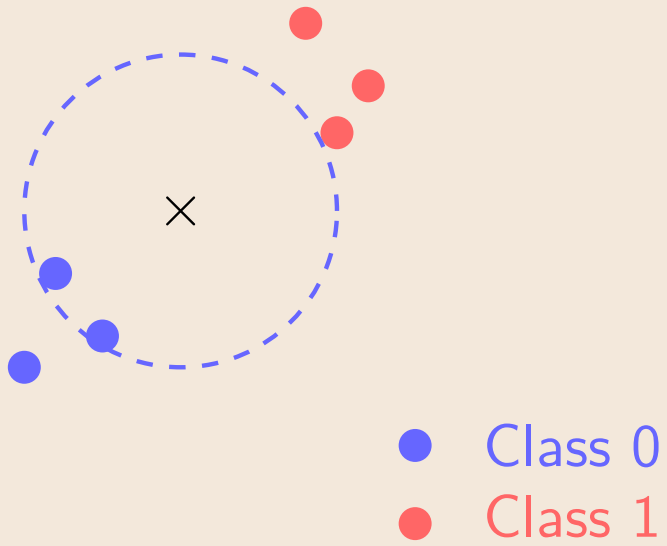
Predict the label of x by majority vote among its k closest neighbors in feature space:

$$\hat{y} = \{y_j : j \in \mathcal{N}_k(x)\},$$

where $\mathcal{N}_k(x)$ denotes the indices of the k nearest neighbors of x.

- Simple, non-parametric method
- No explicit training phase; inference cost depends on dataset size
- Sensitive to distance metric and feature scaling

Visual Illustration: k-Nearest Neighbors (k-NN)



- Dashed circle shows radius containing the k nearest neighbors
- Predict the label for the query based on majority vote within the neighborhood
- Decision depends on chosen distance metric and k

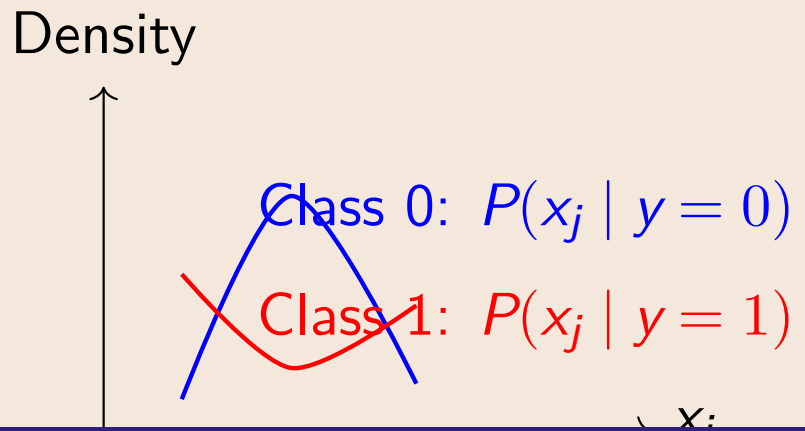
Probabilistic Methods: Naive Bayes Classifier

Key Idea

Predict class using Bayes' rule with the assumption of conditional independence:

$$P(y \mid x) = \frac{P(y) \prod_j P(x_j \mid y)}{P(x)} \quad \propto \quad P(y) \prod_j P(x_j \mid y).$$

Fast, interpretable, effective for high-dimensional or sparse data (e.g., text).



Perceptron: Linear Classifier

Concept

The Perceptron is one of the earliest linear classifiers. It predicts labels based on a weighted sum of inputs:

$$f(x) = \text{sign}\left(w^\top x + b\right)$$

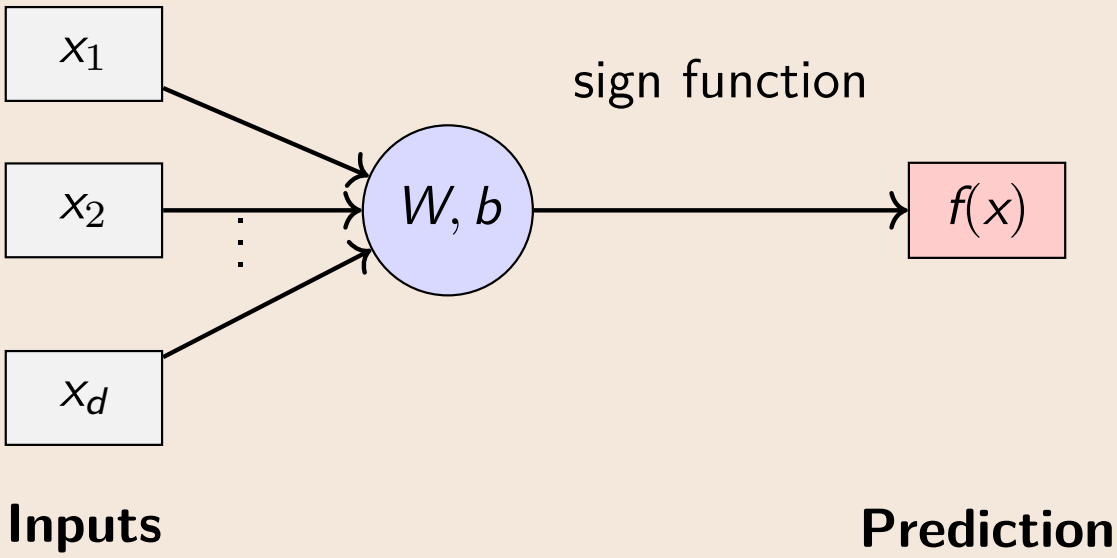
where w are the learned weights, b is the bias term, and $\text{sign}(\cdot)$ outputs -1 or $+1$.

- **Learning Rule:** Iteratively update weights for misclassified examples:

$$w \leftarrow w + \eta\, y_i\, x_i, \quad b \leftarrow b + \eta\, y_i$$

  where η is the learning rate, $y_i \in \{-1, +1\}$.
- **Properties:**
  - Simple, fast, online algorithm
  - Converges if data are linearly separable
  - Cannot solve non-linear problems without transformation

Perceptron: Model Diagram



- Weighted sum: $z = w^\top x + b$
- Activation: $f(x) = \text{sign}(z)$
- Outputs binary class prediction -1 or $+1$

Neural Network Methods: Multi-Layer Perceptron (MLP)

Key Idea

An MLP stacks layers of neurons with learnable weights and nonlinear activations:

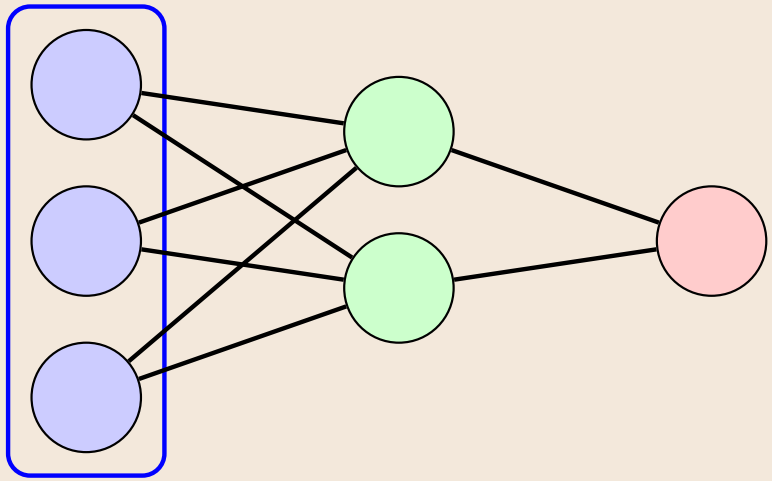
$$h^{(l+1)} = \sigma\left(W^{(l)} h^{(l)} + b^{(l)}\right),$$

where:

- $h^{(0)} = x$ is the input vector
- $h^{(L)}$ is the output (logits or probabilities)
- $\sigma(\cdot)$ applies nonlinearities (e.g., ReLU, sigmoid)

Parameters $W^{(l)}$ and $b^{(l)}$ are optimized end-to-end via backpropagation.

MLP Architecture: Highlighting the Input Layer



Input Layer Hidden Layer Output

Kernel Methods: RBF Kernel SVM

Non-Linear Classification via Kernel Trick

- The RBF (Radial Basis Function) kernel implicitly maps data into a higher-dimensional space:

$$K(x, x') = \exp(-\gamma \|x - x'\|^2)$$

- Enables SVMs to learn non-linear boundaries without explicitly transforming the data.
- Particularly effective when data are not linearly separable in input space.

Kernel Methods: Radial Basis Function (RBF) Kernel

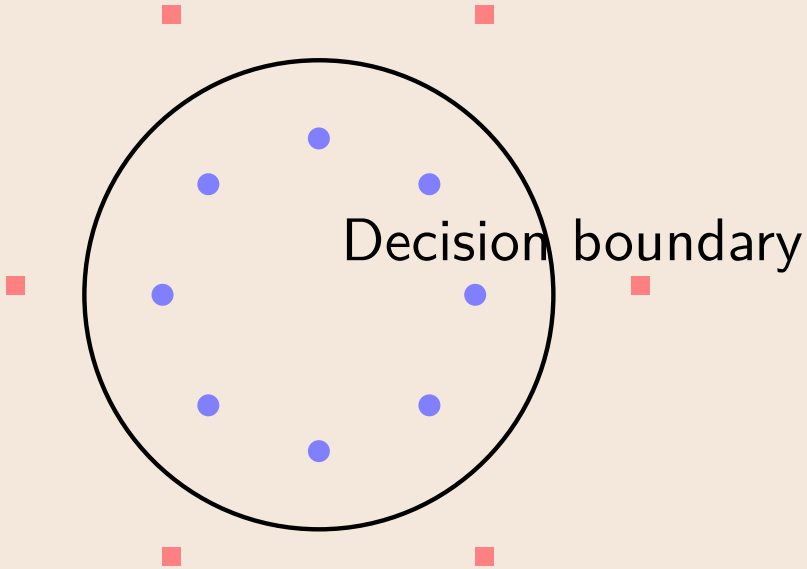
Non-Linear Separation via RBF Kernel

- Defines similarity based on Euclidean distance:

$$K(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$$

- $\gamma > 0$ controls the locality — higher γ creates tighter decision boundaries
- Implicitly maps data into an infinite-dimensional space (via kernel trick)
- Enables SVMs to learn flexible, non-linear decision boundaries without explicit feature engineering

RBF Kernel SVM: Non-Linear Boundary Example



- Circular boundary separates two classes
- Achieved via the RBF kernel without explicit feature engineering
- Effective for non-linearly separable problems

Supervised Learning: Evaluation

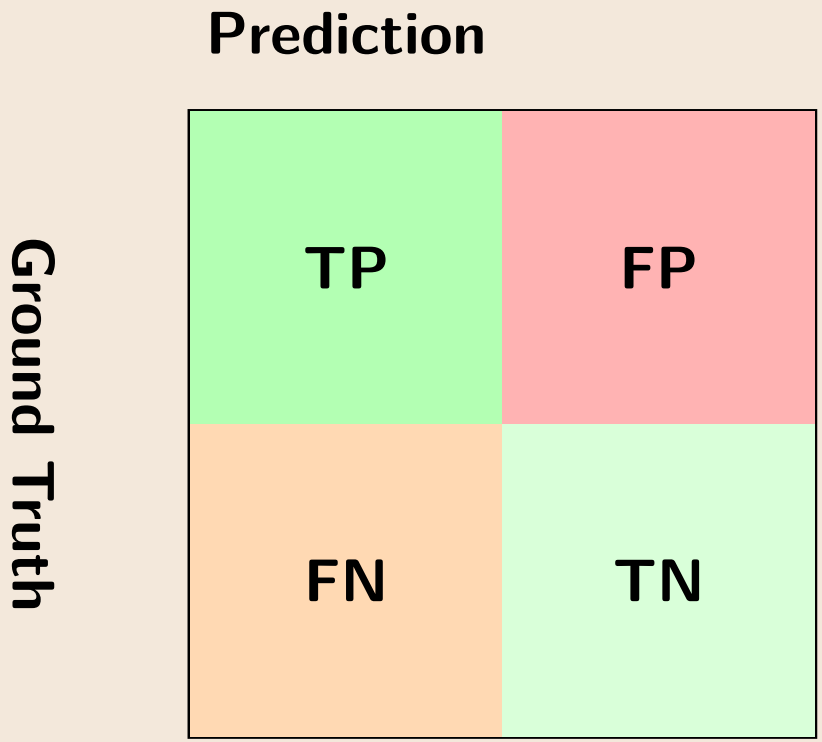
Why Evaluation?

- Ensures models generalize to unseen data
- Quantifies performance for model selection
- Guides tuning of hyperparameters and features

Common Evaluation Dimensions

- **Classification Tasks**: Accuracy, Precision, Recall, F1 Score, ROC AUC, Confusion Matrix
- **Regression Tasks**: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R^2 (coefficient of determination)
- **Model Robustness**: Performance on noisy or imbalanced datasets
- **Interpretability & Fairness**: Understandable decisions, bias mitigation

Confusion Matrix (Binary)



Supervised Learning: Core Evaluation Metrics

Confusion Matrix Terms

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

Common Metrics

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad \text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \quad F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

ROC Curve & AUC: Evaluating Classifier Performance

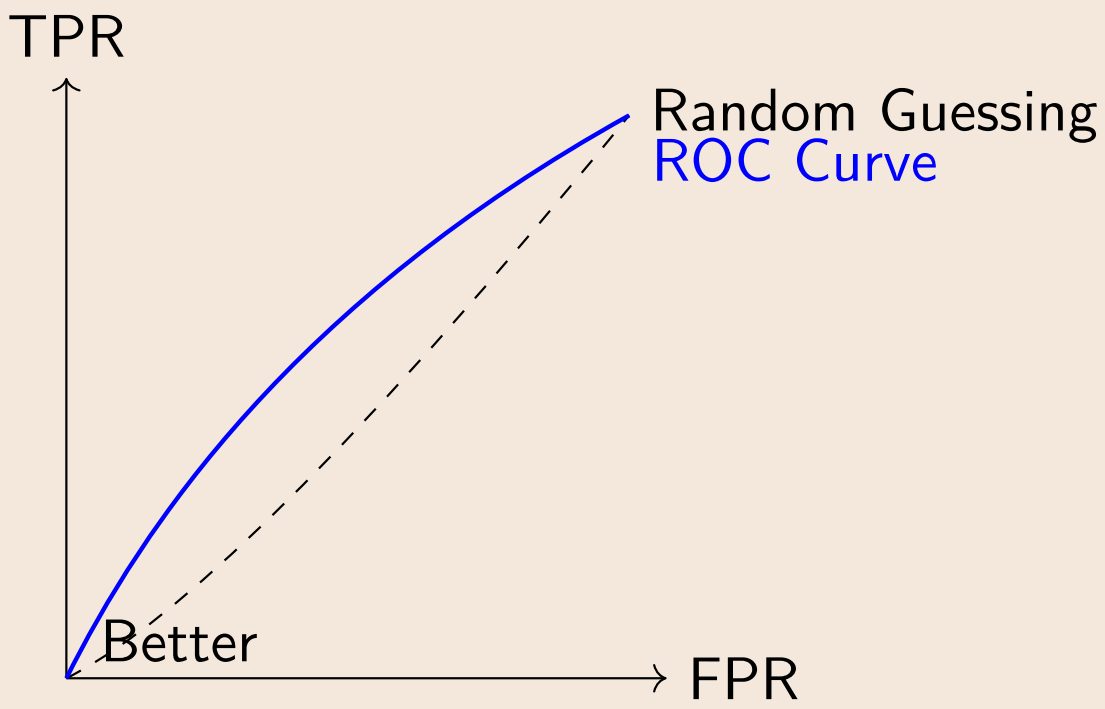
ROC Curve

Plots the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) at various thresholds.

$$\text{ROC AUC} = \int_0^1 \text{TPR}\left(FPR^{-1}(u)\right)\, du$$

- AUC = Area Under the Curve
- AUC close to 1.0 indicates strong classifier performance
- AUC = 0.5 suggests random guessing

ROC Curve: Visual Example



Unsupervised Learning

Formalization and overview

Unsupervised Learning: Introduction

Key Idea

Unsupervised learning discovers hidden patterns or structures in unlabeled data.

- No ground-truth labels provided
- The model learns to group, structure, or summarize the data
- Enables knowledge discovery from raw datasets

Common Applications:

- Customer segmentation
- Anomaly detection
- Dimensionality reduction and visualization

Unsupervised Learning: Formalization

General Setup

- Given dataset:
$$\mathcal{D} = \{x_1, x_2, \ldots, x_N\}, \quad x_i \in \mathbb{R}^d$$
- No labels y_i are observed
- Goal: Find structure, groups, or representations of $\mathcal{D}$

Types of Structure Learned:

- Clusters or groups of similar data points
- Lower-dimensional embeddings of data
- Detection of unusual or anomalous points

Unsupervised Learning: Techniques Overview

- **Clustering Algorithms** Group data points into distinct clusters based on similarity Examples: k-Means, Hierarchical Clustering, DBSCAN
- **Dimensionality Reduction** Compress high-dimensional data while preserving structure Examples: Principal Component Analysis (PCA), t-SNE, UMAP
- **Density Estimation & Anomaly Detection** Estimate data distribution to identify rare or abnormal cases Examples: Gaussian Mixture Models (GMM), One-Class SVM

k-Means Clustering: Concept & Objective

Goal

Partition N data points into K clusters to minimize within-cluster variance:

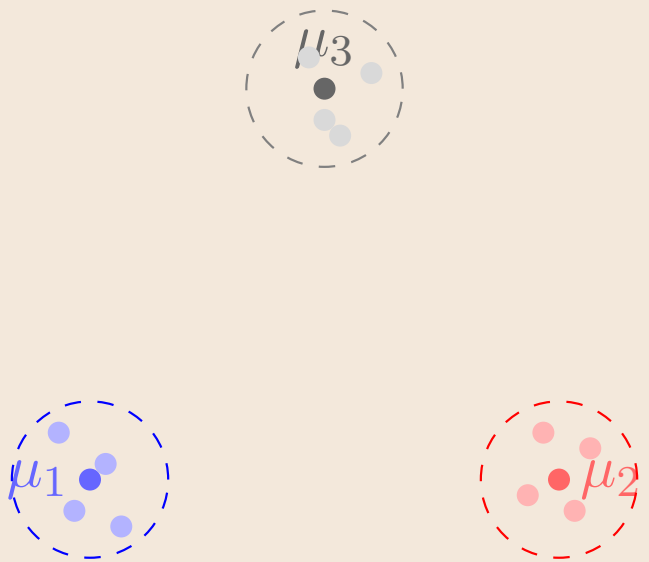
$$\min_{\{\mathcal{C}_k\},\{\mu_k\}} \sum_{k=1}^{K} \sum_{x_i \in \mathcal{C}_k} \|x_i - \mu_k\|^2$$

where:

- $\mathcal{C}_k$ = set of points assigned to cluster k
- μ_k = centroid (mean) of cluster k

- Alternates between point assignment and centroid update
- Assumes spherical clusters with equal variance
- Sensitive to initialization and choice of K

k-Means Clustering: Visual Example



- Points grouped by proximity to nearest centroid
- Centroids updated iteratively to minimize total squared distances

Other Learning Paradigms

- **Semi-supervised**: leverages both labeled and unlabeled data
- **Reinforcement**: agents learn via rewards in environments
- **Self-supervised**: constructs its own labels (e.g. masked language modeling)
- **Online/Continual**: adapts to streaming, non-stationary data
- **Meta-learning & Few-shot**: learns to learn from few examples

Semi-Supervised Learning (SSL)

Key Idea

Combines a small amount of labeled data with a large amount of unlabeled data to improve learning performance.

- Useful when labels are expensive but unlabeled data is abundant
- Bridges the gap between supervised and unsupervised learning
- Common techniques: self-training, pseudo-labeling, consistency regularization

Reinforcement Learning (RL)

Learning by Interaction

An agent learns to make decisions by interacting with an environment to maximize cumulative reward.

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

- π = policy mapping states to actions
- γ = discount factor for future rewards
- Used in robotics, games, autonomous systems

Self-Supervised Learning (SSL)

Learning from Pretext Tasks

Models create their own labels from raw data by solving auxiliary tasks that encourage meaningful representations.

- Predict missing data (e.g., masked language modeling)
- Contrastive learning to distinguish similar vs. dissimilar examples
- Foundation for modern language and vision models (e.g., BERT, SimCLR)

Federated Learning

Decentralized Collaborative Learning

Multiple devices collaboratively train a model without sharing raw data, preserving privacy.

- Each client trains locally and shares model updates
- Aggregation (e.g., Federated Averaging) combines updates centrally
- Applications: mobile devices, healthcare, IoT

Online & Continual Learning

Learning Over Time

Models continuously update from new data streams, aiming to avoid catastrophic forgetting.

- Online learning: updates with each new sample or batch
- Continual learning: maintains performance on past tasks while learning new ones
- Critical for dynamic environments and real-time applications

Meta-Learning & Few-Shot Learning

Learning to Learn

Meta-learning focuses on building models that quickly adapt to new tasks with minimal data.

- Few-shot learning: Generalize from very few examples per class
- Meta-learning trains across tasks to extract transferable knowledge
- Popular approaches: Model-Agnostic Meta-Learning (MAML), Prototypical Networks

Transfer Learning

Knowledge Transfer Across Tasks

Reuses knowledge from a source task or domain to improve performance on a related target task.

- Reduces need for large labeled datasets in new tasks
- Common in deep learning: fine-tuning pre-trained models (e.g., ImageNet, BERT)
- Enables faster convergence and improved generalization

Multi-Task Learning (MTL)

Joint Learning Across Tasks

Simultaneously trains a model on multiple related tasks to leverage shared structure.

- Improves generalization through inductive transfer
- Example: Shared encoder with task-specific output heads
- Reduces overfitting and enhances efficiency

Multi-Modal Learning

Integrating Multiple Data Modalities

Combines information from different modalities (e.g., text, images, audio) for richer understanding.

- Enables including like image captioning, video understanding, and vision-language reasoning
- Foundation for multimodal models (e.g., CLIP, DALL-E, GPT-4 Vision)
- Aligns heterogeneous data in joint representation spaces

Active Learning

Efficient Labeling via Selective Querying

The model iteratively selects the most informative unlabeled examples for annotation.

- Reduces labeling costs by focusing on uncertain or representative samples
- Popular strategies: uncertainty sampling, query-by-committee, diversity sampling
- Useful for domains with expensive annotations (e.g., medical imaging)

Large Language Models (LLMs)

Foundation Models for Language Tasks

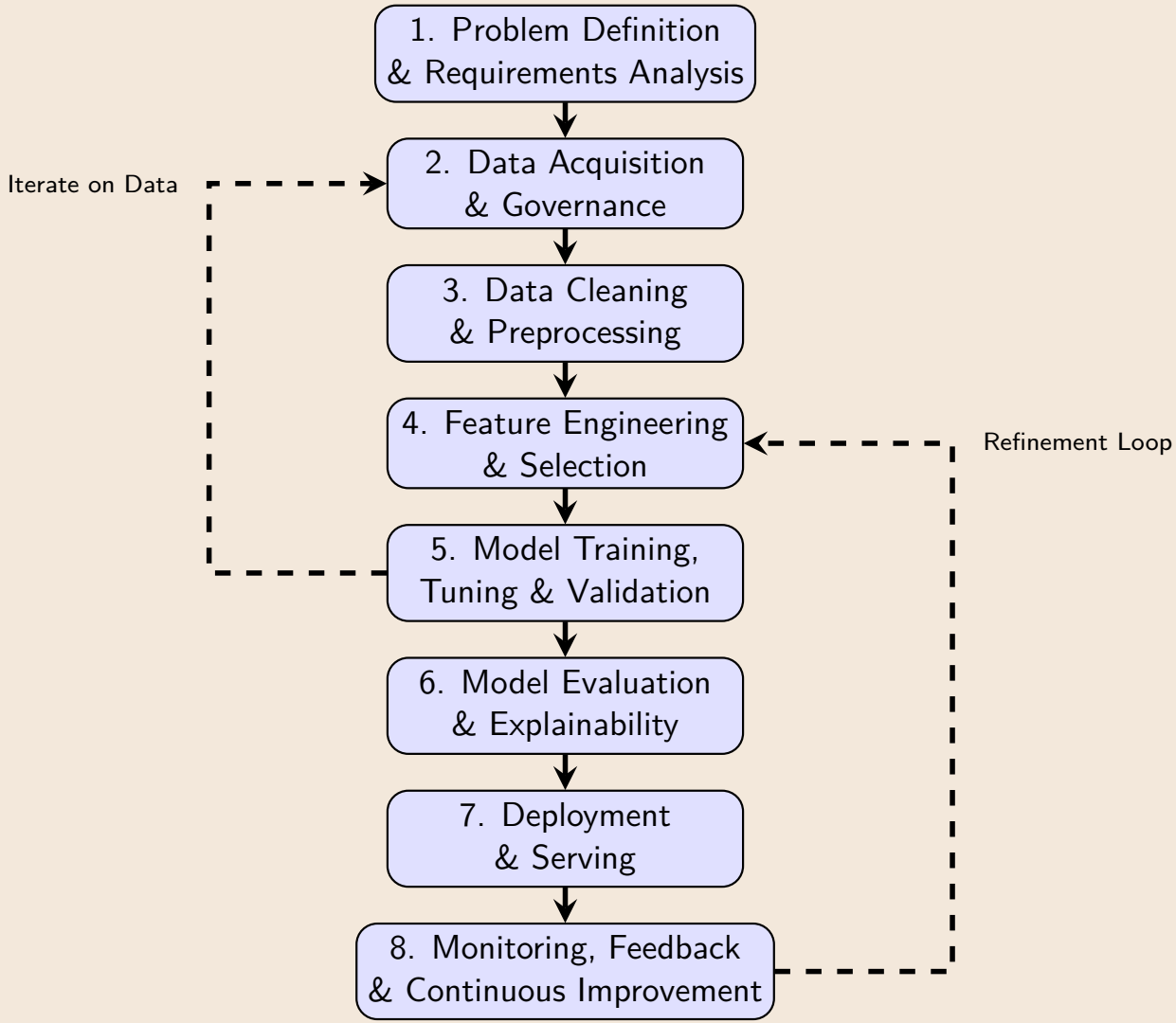
LLMs are massive transformer-based models pre-trained on vast corpora, capable of diverse language understanding and generation.

- Pre-training with self-supervised objectives (e.g., masked tokens, next-word prediction)
- Fine-tuned for downstream tasks: QA, summarization, dialogue, code
- Examples: GPT-3/4, BERT, PaLM, LLaMA
- Serve as building blocks for modern AI applications

MLOPs & End-to-End Machine Learning

Building Production-grade ML systems

End-to-End Machine Learning Workflow (Visual Overview)



Iterative loops reflect real-world model improvement cycles driven by monitoring and

End-to-End Machine Learning Workflow

ML Lifecycle: From Data to Production

Building reliable, production-grade ML systems requires coordinated stages beyond just model development:

- **Problem Definition & Requirements Analysis**
- **Data Acquisition and Governance**
- **Data Cleaning & Preprocessing**
- **Feature Engineering and Selection**
- **Model Training, Tuning, and Validation**
- **Model Evaluation & Explainability**
- **Deployment & Serving Infrastructure**
- **Monitoring, Feedback Loops, and Continuous Improvement**

Challenges in End-to-End ML Pipelines

Key Technical and Organizational Challenges

Successful deployment of ML systems at scale requires addressing multiple real-world obstacles:

- **Data Drift:** Evolving data distributions reduce model reliability over time
- **Model Degradation:** Performance decay under changing real-world conditions
- **Reproducibility Gaps:** Inconsistent results across environments or pipelines
- **Scalability Constraints:** Bottlenecks in large-scale training and serving infrastructure
- **Cross-Functional Collaboration:** Bridging gaps between data science, engineering, and operations teams
- **Monitoring and Alerting:** Detecting silent model failures and anomalies in production

MLOps: Operationalizing Machine Learning

What is MLOps?

MLOps (Machine Learning Operations) applies DevOps principles to automate, scale, and govern the end-to-end ML lifecycle—from experimentation to reliable production deployment.

- **CI/CD for ML:** Automated pipelines for code, data, and model integration and deployment
- **Version Control:** Tracking of code, datasets, features, and trained models
- **Testing and Validation:** Automated tests for model performance, bias, and robustness
- **Monitoring and Observability:** Detecting data drift, performance decay, and anomalies in production
- **Governance and Compliance:** Ensuring auditability, explainability, and regulatory alignment

MLOps: Core Components for Scalable ML Systems

- **Data Versioning:** Systematic tracking of datasets, features, and schema changes
- **Model Registry:** Centralized catalog for storing, versioning, and promoting trained models
- **Pipeline Automation:** Orchestration of reproducible workflows for training, evaluation, and deployment
- **Monitoring & Observability:** Real-time tracking of model performance, data drift, and system health
- **Infrastructure Management:** Scalable, fault-tolerant platforms for model serving and resource provisioning

Goal

Deliver reliable, maintainable, and auditable ML solutions that perform consistently in production environments.

Technologies & Tools Across the ML Lifecycle

Lifecycle Stage	Representative Tools and Technologies
Problem Definition & Planning	Jupyter, Google Colab, Miro, Notion, Confluence
Data Acquisition & Integration	SQL, Apache Kafka, Airflow, Databricks, Snowflake
Exploratory Data Analysis (EDA)	pandas, NumPy, matplotlib, seaborn, Tableau, Power BI
Feature Engineering & Processing	scikit-learn, Featuretools, TFX, PySpark, dbt
Model Selection & Experimentation	scikit-learn, XGBoost, LightGBM, Optuna, Weights & Biases
Model Training & Optimization	TensorFlow, PyTorch, Keras, SageMaker, Ray
Validation & Evaluation	MLflow, TensorBoard, Fairlearn, Alibi, SHAP
Model Deployment & Serving	Docker, Kubernetes, BentoML, Seldon Core, TorchServe
Monitoring & Observability	Prometheus, Grafana, Evidently, Arize, Fiddler AI
Governance & Documentation	Model Cards, Datasheets for Datasets, Git, MLflow Tracking

Key Roles Across the ML Lifecycle

Role	Primary Responsibilities
Data Engineer	Design, build, and maintain data pipelines, ingestion processes, transformations, and enforce data governance standards
Data Scientist	Define ML problems, conduct exploratory data analysis (EDA), build prototypes, perform feature engineering, and evaluate models
Feature Engineer	Create, manage, and ensure the quality, consistency, and reusability of features used during training and inference
ML Engineer	Productionize ML models, optimize pipelines, manage deployments, and integrate models with application systems
MLOps Engineer	Automate CI/CD pipelines, manage serving infrastructure, implement monitoring, and ensure system reliability and compliance

ML Unit Wrap-Up: Reflections & Emerging Trends

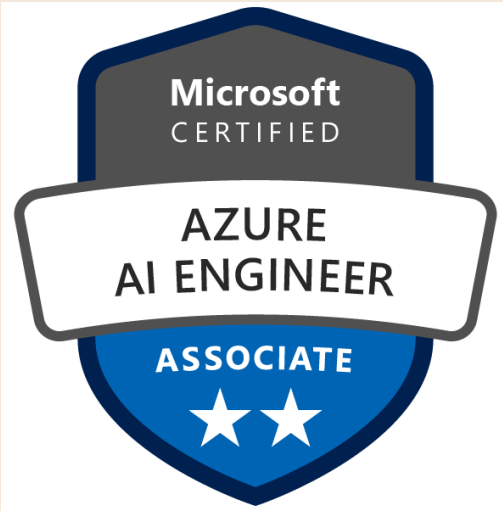
Key Themes to Retain

The machine learning landscape is evolving beyond core algorithms toward production-grade, responsible, and adaptable systems.

- **Hybrid Modeling:** Combining symbolic reasoning with neural networks; rise of multi-modal architectures
- **Automated MLOps:** Scalable CI/CD, drift detection, automated retraining pipelines for reliable ML in production
- **Responsible AI:** Focus on fairness audits, transparency, privacy by design, and energy-efficient "Green AI"
- **Foundation & Self-Supervised Models:** General-purpose pre-trained models accelerating downstream tasks
- **Cross-Disciplinary Integration:** Tighter convergence of ML with domain sciences, ethics, and socio-technical considerations

Industry Certificates

AWS AI Cloud Practitioner



Microsoft Certified: Azure AI Engineer

Recommended Books

- **AI Engineering: Building Applications with Foundation Models**
  Chip Huyen, 1st Edition, January 2025
- **LLM Engineer's Handbook: Master the Art of Engineering Large Language Models from Concept to Production**
  Paul Iusztin *et al.*, October 2024
- **Transformers for Natural Language Processing and Computer Vision**
  Denis Rothman, 3rd Edition, February 2024

Module 1 Conclusion

- **Artificial Intelligence (AI):** Broad field of systems capable of perception, reasoning, learning, and decision-making
- **Machine Learning (ML):** Subset of AI focused on data-driven model development and prediction
- **Learning Paradigms:** Supervised, unsupervised, semi-supervised, self-supervised, reinforcement learning, and beyond
- **ML Lifecycle:** End-to-end workflows from problem framing to deployment, monitoring, and continuous improvement
- **MLOps:** Operational practices enabling scalable, reliable, and governed ML in production environments
- **Emerging Trends:** Neuro-symbolic AI, multi-modal systems, responsible AI, and foundation models shaping future developments