**Artificial Intelligence (AI)** is the broad field focused on creating systems that can perform tasks typically requiring human intelligence - like reasoning, perception, or decision-making.

**Machine Learning (ML)** is a subset of AI where systems learn patterns from data rather than being explicitly programmed. Instead of writing specific rules, we feed algorithms data and let them discover patterns automatically.

# Types of Machine Learning

## Supervised Learning

The algorithm learns from labeled training data - you provide both inputs and the correct outputs.

**Classification**: Predicting categories (spam vs. not spam emails, cancer vs. benign tissue)
**Regression**: Predicting continuous values (protein concentration, gene expression levels)

## Unsupervised Learning

The algorithm finds hidden patterns in data without labeled examples.

**Clustering**: Grouping similar data points (identifying cell types from gene expression profiles) **Dimensionality Reduction**: Simplifying data while preserving important information (visualizing high-dimensional genomic data)

## Reinforcement Learning

The algorithm learns through trial and error, receiving rewards or penalties for actions. Think of it like training a system to optimize experimental conditions by testing different parameters and learning from results.

## Semi-supervised Learning

Combines small amounts of labeled data with lots of unlabeled data - useful when labeling is expensive or time-consuming.

# The Machine Learning Pipeline

## 1. Problem Definition

- What question are you trying to answer?
- Is this a classification, regression, or clustering problem?
- What would success look like?

## 2. Data Collection & Preparation

- Gathering relevant datasets
- Clean the data (handle missing values, remove outliers)
- Feature engineering (creating meaningful variables from raw data)

### 3. Exploratory Data Analysis

- Visualize data distributions
- Look for correlations and patterns
- Understand data quality and potential biases

### 4. Model Selection & Training

- Choose appropriate algorithms based on your problem type
- Split data into training, validation, and test sets
- Train multiple models and compare performance

### 5. Model Evaluation

- Use metrics appropriate to your problem (accuracy, precision, recall for classification; $R^2$ for regression)
- Cross-validation to ensure the model generalizes well
- Test on completely unseen data

### 6. Model Deployment & Monitoring

- Implement the model in your workflow
- Monitor performance over time
- Retrain as needed with new data

## Key Concepts

**Overfitting**: When a model memorizes training data but fails on new data.

**Bias-Variance Tradeoff**: Balancing model complexity. Too simple (high bias) misses important patterns; too complex (high variance) doesn't generalize well.

**Feature Selection**: Choosing which variables to include. This means selecting which attributes are most informative for your analysis.

## Types of Classifiers

### Linear Classifiers

**Logistic Regression** Uses a linear decision boundary to separate classes. Despite the name, it's used for classification. Simple, interpretable, and works well when classes are roughly linearly separable.

**Support Vector Machines (SVM)** Finds the optimal boundary (hyperplane) that maximally separates classes. Can handle non-linear data using kernel tricks. Effective in high-dimensional spaces.

**Naive Bayes** Based on Bayes' theorem, assumes features are independent. Fast and works well with small datasets.

### Tree-Based Classifiers

**Decision Trees** Creates a flowchart-like structure of if-then rules. Highly interpretable but prone to overfitting. Easy to understand and explain to others.

**Random Forest** Combines many decision trees and averages their predictions. Reduces overfitting while maintaining good performance. Handles missing values well.

**Gradient Boosting (XGBoost, LightGBM)** Builds models sequentially, each correcting errors from previous ones. Often achieves excellent performance in competitions but can be complex to tune.

### Instance-Based Classifiers

**k-Nearest Neighbors (k-NN)** Classifies based on the majority class of k nearest neighbors. Simple concept but can be computationally expensive. No explicit training phase - "lazy learning."

### Neural Network Classifiers

**Multi-Layer Perceptron (MLP)** Basic neural network with hidden layers. Can learn complex non-linear patterns but requires more data and tuning than simpler methods.

**Deep Neural Networks** Multiple hidden layers that can learn hierarchical features. Excellent for complex data like images, text, or audio but requires substantial computational resources.

### Probabilistic Classifiers

**Gaussian Mixture Models** Assumes data comes from a mixture of Gaussian distributions. Provides probability estimates for class membership.

## Choosing the Right Classifier

**Linear methods** work well when you have many features relative to samples, or when classes are linearly separable.

**Tree-based methods** excel with mixed data types, non-linear relationships, and when interpretability matters.

**Neural networks** shine with large datasets and complex patterns, especially in images, text, or sequential data.

**k-NN** works well when local patterns matter and you have sufficient data density.

The best approach often involves trying multiple classifiers and using cross-validation to compare their performance on your specific dataset and problem.