Machine Learning Reference Guide

Model Selection, Evaluation & Decision Framework

A comprehensive technical field manual covering model families, algorithm trade-offs, and advanced considerations for senior data scientists and machine learning engineers.

Esmaeil Rezaei, Ph.D.

Document Scope

This document provides structured, technically dense coverage of:

17 major algorithm categories · 80+ methods and architectures

Each method is analyzed through four lenses: Pros · Cons · Technical Use Case · Challenges

Intended as: *Decision Framework* | *Interview Reference* | *Technical Field Manual*

Version 1.0 · ML Reference Series

Contents

Regression Models

1.1 Ordinary Least Squares (OLS) Linear Regression

Ordinary Least Squares regression is the foundational workhorse of statistical estimation, introduced formally by Legendre (1805) and Gauss (1809) to solve overdetermined systems of linear equations arising in astronomical measurement. The core motivation was to find the unique linear combination of predictors that minimizes the sum of squared residuals—a criterion chosen for its analytical tractability and its connection to maximum likelihood estimation under Gaussian errors. OLS remains the canonical starting point for regression analysis because of the Gauss-Markov theorem, which establishes that the OLS estimator is the Best Linear Unbiased Estimator (BLUE) under mild conditions, making it a theoretical benchmark against which all other regression methods are measured.

✓ Pros

- Closed-form solution guarantees global optimum with no convergence issues.
- Perfectly interpretable: coefficients are partial derivatives of the response with respect to each feature.
- BLUE (Best Linear Unbiased Estimator) under Gauss-Markov assumptions.
- Computationally trivial for modest n, p; well-understood statistical inference (confidence intervals, hypothesis tests, F-tests).
- Residual diagnostics are mature and actionable.

✗ Cons

- Assumes linearity, homoscedasticity, independence, and normality of residuals; violations degrade inference.
- Singular or near-singular ($X^T X$) causes numerical instability or non-identifiability when $p \approx n$ or multicollinearity exists.
- No implicit regularization; severe overfitting in high-dimensional settings ($p > n$).
- Highly sensitive to outliers (squared loss is non-robust).
- Cannot model non-linear relationships without manual feature engineering.

▷ Technical Use Case

Use when the data-generating process is plausibly linear, $n \gg p$, features are not strongly collinear, and interpretability of coefficients with statistical inference is required. Appropriate when the hypothesis space geometry is a hyperplane and the optimization landscape convexity is exploitable analytically. Not appropriate in high-dimensional sparse settings or when distributional assumptions are clearly violated.

★ Challenges & Advanced Considerations

- Multicollinearity inflates coefficient variance without affecting predictions; VIF analysis required.
- Heteroscedasticity invalidates standard errors; must use White/HC robust SEs or WLS.
- Omitted variable bias is silent and structurally undetectable without domain knowledge.
- Leverage points versus high-influence points: Cook's D, DFFITS diagnostic gaps often ignored in practice.
- *Interview-level:* Under which exact conditions is OLS equivalent to MLE? What are the implications when the true error distribution is heavy-tailed?

Variance Inflation Factor (VIF):

- VIF measures multicollinearity between predictors.
- Formula for VIF for feature j:

$$\text{VIF}_j = \frac{1}{1 - R_j^2}$$

  where R_j^2 is the R^2 from regressing feature j on all other features.
- This formula can be computed for each feature individually or for all features.
- **Solution for high VIF:** Remove or combine highly collinear features, or use regularization (Ridge/Lasso).

White / Heteroskedasticity-Consistent (HC) Standard Errors:

- If variance of errors is not constant, we can use White/HC formula to estimate variance correctly.
- Changing variance estimates does not affect coefficient estimates.
- **Benefit:** Correct statistical inference.
- OLS uses variance only for:
  - Standard errors
  - t-tests
  - p-values
  - Confidence intervals
- OLS does NOT use variance for:
  - Coefficient estimation
  - Prediction
  - Model fitting

Omitted Variable:

- A variable that may contribute to the model but is not included.
- Example:

$$Y = \beta X + \gamma Z + \epsilon$$

  where Z is omitted.
- In almost every real-world ML model, there are omitted variables.
- **Omitted Variable Bias (OVB)** occurs if:

$$\mathbb{E}[\hat{\beta}] \neq \beta$$

  Meaning: On average, your estimated coefficient does not equal the true effect.
- **Important points:**
  - Large R^2 does NOT protect against OVB.
  - R^2 does not tell you whether coefficients are unbiased.
  - If the goal is prediction, OVB is less critical.
  - If the goal is causal inference, address OVB by including the variable or using a correlated proxy.

Leverage and Influence:

- **Leverage point:** Observation with extreme values in predictor space (X-space).

$$h_{ii} = x_i(X'X)^{-1}x_i'$$

- **Cook's Distance (Cook's D):** Measures how much all regression coefficients change if observation i is removed.
- **DFFITS:** Measures how much the predicted value for observation i changes if i is removed.

- **Why diagnostic gaps are often ignored:**
  - Focus on prediction accuracy.
  - Use cross-validation.
  - Use tree-based or robust models.
  - Influence diagnostics are skipped because:
    - Influence matters more in causal contexts.
    - Large datasets reduce effect of one observation.
    - Automated pipelines rarely include these checks.
- **When ignoring is dangerous:**
  - Small sample size
  - Coefficient interpretation matters
  - Data may contain outliers
  - Policy analysis or research
- **Summary:**
  - Leverage → extreme X values
  - Influence → changes model if removed
  - Cook's D → global parameter change
  - DFFITS → change in predicted value

Interview-level: OLS vs. MLE:

- OLS is equivalent to Maximum Likelihood Estimation (MLE) under the following conditions:
  - Linear model specification
  - Errors are independent and identically distributed (i.i.d.)
  - Errors are normally distributed with zero mean and constant variance
- **Implications for heavy-tailed error distributions:**
  - OLS estimates remain unbiased if other assumptions hold.
  - Standard errors, t-tests, and confidence intervals may be invalid.
  - Robust methods or MLE with appropriate error distribution may be preferred.

1.2 Ridge Regression (L_2 Regularization)

Ridge regression was proposed by Hoerl and Kennard (1970) to address a fundamental breakdown of OLS: when predictor variables are highly collinear, the matrix $(X^\top X)$ becomes near-singular and its inversion produces wildly unstable, high-variance coefficient estimates. The proposed remedy was to add a small positive constant λ to the diagonal of $(X^\top X)$ before inversion—a technique known as Tikhonov regularization in the numerical analysis literature. This L_2 penalty continuously shrinks all coefficients toward zero, **trading a small amount of bias for a large reduction in variance**, and represents the first systematic application of the bias-variance tradeoff as a deliberate modeling strategy.

✓ Pros

- Stabilizes $(X^T X + \lambda I)^{-1}$; always invertible for $\lambda > 0$, eliminating multicollinearity breakdown.
- Continuously shrinks all coefficients toward zero; variance reduction with modest bias increase.
- Closed-form solution retained; computationally efficient.
- Differentiable everywhere; gradient-based optimization is straightforward.
- Consistent predictor when true model is dense (many small effects).

✗ Cons

- Does not produce sparse solutions; cannot perform variable selection.
- Requires tuning of λ via cross-validation; adds a layer of model selection complexity.
- Coefficients are biased estimators; invalid for causal inference.
- Sensitive to feature scaling; mandatory standardization before fitting.
- In $p \gg n$ settings, does not handle true sparsity as efficiently as L_1.

▷ Technical Use Case

Optimal when the true signal is believed to be distributed across many features (dense signal), features are correlated, and prediction accuracy is prioritized over exact sparsity. Use in $n < p$ scenarios where OLS fails. The bias-variance tradeoff favors Ridge when variance dominates MSE. Convex optimization landscape ensures global minimum.

★ Challenges & Advanced Considerations

- Choice of λ range for grid search is non-trivial; log-scale grids over many orders of magnitude required.
- Interpretation of shrunk coefficients loses the partial-effect semantics of OLS.
- Regularization path analysis via SVD reveals that Ridge shrinks more along directions of low variance in X; often misunderstood.
- *Interview-level:* Derive the ridge estimator as a MAP estimate under a Gaussian prior. What prior corresponds to Lasso?

L2 (Ridge) Closed-Form Solution:

- Saying "L2 still has a closed-form solution" means we can write the exact optimizer as an explicit formula — no iterative search needed.
- **Start with OLS:**
$$\hat{\beta}_{OLS} = (X^\top X)^{-1} X^\top y$$
- **Add L2 penalty:**
$$\min_\beta (y - X\beta)^\top (y - X\beta) + \lambda \|\beta\|_2^2$$
- Take derivative, set to zero:
$$X^\top X\beta - X^\top y + \lambda\beta = 0$$
- Rearrange:
$$(X^\top X + \lambda I)\beta = X^\top y$$
- So the Ridge solution is:
$$\hat{\beta}_{ridge} = (X^\top X + \lambda I)^{-1} X^\top y$$
- Still closed-form. Convex. Unique solution.

Ridge Regularization Path Analysis via SVD:

1. **SVD of X:**
$$X = U\Sigma V^\top, \quad \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r), \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$$
Singular values σ_j measure variance along principal directions.
2. **OLS in SVD coordinates:**
$$\hat{\beta}_{OLS} = V\Sigma^{-1} U^\top y$$
Each component scaled by $1/\sigma_j$. Small $\sigma_j \to$ instability.

3. **Ridge in SVD form:**
$$\hat{\beta}_{ridge} = V\,\mathrm{diag}\Big(\frac{\sigma_j}{\sigma_j^2+\lambda}\Big)U^\top y$$
Shrinkage factor: $\frac{\sigma_j}{\sigma_j^2+\lambda}$
4. **High-variance directions:** $\sigma_j^2 \gg \lambda \implies$ barely shrunk, almost like OLS.
5. **Low-variance directions:** $\sigma_j^2 \ll \lambda \implies$ heavily shrunk.
6. **Misunderstanding:** Ridge does not shrink all coefficients uniformly toward zero. It shrinks more along directions with low information (small σ_j).
7. **Low variance problem:**
   - Small $\sigma_j \to 1/\sigma_j$ becomes huge in OLS.
   - Tiny noise in y gets amplified $\to$ coefficient instability.
   - Geometric picture: data almost flat along that direction; OLS slope can swing wildly, predictions remain similar, coefficients explode.

Ridge as a MAP Estimator:

- MAP = Maximum A Posteriori:
$$\hat{\theta}_{MAP} = \arg\max_\theta p(\theta \mid \text{data})$$
Using Bayes' theorem:
$$p(\theta \mid y) \propto p(y \mid \theta)p(\theta)$$
- Log form:
$$\hat{\theta}_{MAP} = \arg\max_\theta \log p(y \mid \theta) + \log p(\theta)$$
- **Step 1: Likelihood (Linear Model)**
$$y = X\beta + \epsilon, \quad \epsilon \sim N(0, \sigma^2 I)$$
$$p(y \mid \beta) \propto \exp\Big(-\frac{1}{2\sigma^2}\|y - X\beta\|_2^2\Big)$$
- **Step 2: Prior on β (Ridge)**
$$\beta \sim N(0, \tau^2 I), \quad p(\beta) \propto \exp\Big(-\frac{1}{2\tau^2}\|\beta\|_2^2\Big)$$
- **Step 3: Posterior**
$$p(\beta \mid y) \propto p(y \mid \beta)p(\beta) \propto \exp\Big(-\frac{1}{2\sigma^2}\|y - X\beta\|_2^2 - \frac{1}{2\tau^2}\|\beta\|_2^2\Big)$$
- **Step 4: MAP Estimate**
$$\hat{\beta}_{MAP} = \arg\min_\beta \frac{1}{2\sigma^2}\|y - X\beta\|_2^2 + \frac{1}{2\tau^2}\|\beta\|_2^2$$
Multiply through by $2\sigma^2$:
$$\min_\beta \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2, \quad \lambda = \frac{\sigma^2}{\tau^2}$$
- This is exactly Ridge regression.
- **Prior for Lasso:** Laplace (double-exponential) prior on β leads to L1 penalty.

1.3 Lasso Regression (L_1 Regularization)

The Lasso (Least Absolute Shrinkage and Selection Operator) was introduced by Tibshirani (1996) to overcome a key limitation of Ridge regression: while Ridge shrinks coefficients toward zero, it never sets them exactly to zero and therefore cannot perform variable selection. The Lasso replaces the L_2 penalty with an L_1 penalty on the coefficient vector, whose non-differentiable geometry at the origin creates a sparse solution where many coefficients are exactly zero. This simultaneous estimation and selection property was a major conceptual advance, enabling analysts to work with high-dimensional data ($p \gg n$) and produce interpretable, parsimonious models without a separate variable selection step.

✓ Pros

- Induces exact sparsity; performs simultaneous variable selection and estimation.
- Scales well to very high-dimensional problems when the true signal is sparse.
- Interpretable sparse model; coefficients of irrelevant features become exactly zero.
- Strong theoretical guarantees under restricted isometry property (RIP) and irrepresentability conditions.

✗ Cons

- Not differentiable at zero; requires subgradient or coordinate descent optimization.
- Among correlated predictors, arbitrarily selects one; unstable variable selection.
- Can select at most $\min(n, p)$ features; inconsistent in large-p dense scenarios.
- Coefficient paths are piecewise linear but non-monotone; solution not unique when $p > n$.
- Shrinkage of truly non-zero coefficients introduces bias that does not vanish asymptotically.

▷ Technical Use Case

Use when the true model is sparse: few features have non-zero effects. Ideal for $p \gg n$ settings requiring dimensionality reduction embedded in estimation. Inappropriate when many features contribute small-but-nonzero effects (dense signal). Convex but non-smooth; coordinate descent is the standard computational approach.

★ Challenges & Advanced Considerations

- Variable selection instability in the presence of correlated features is a fundamental problem; stability selection or bootstrapped Lasso paths help.
- Post-selection inference (PSI) is non-trivial: selecting variables then performing standard inference leads to invalid p-values.
- Irrepresentability condition for sign-consistent variable selection is often violated in practice.
- *Interview-level:* Under what conditions does Lasso achieve oracle properties? Compare Lasso vs. Adaptive Lasso consistency.

L1 (Lasso) in High-Dimensional Problems:

- L1 scales well when the true signal is sparse.
- Important: First handle correlations via feature selection before applying Lasso.
  - Otherwise, we face instability or unclear/fragile solutions.
- **Coefficient paths:** Piecewise linear as a function of λ, but non-monotone.
  - $\beta(\lambda)$ increases linearly within intervals.
  - Slopes change at "kinks" when variables enter or leave the model.
  - LARS (Least Angle Regression) efficiently computes the full Lasso path by jumping from kink to kink.
  - Recommendation: Combine Lasso with filtering techniques to handle high-correlation features.

Bootstrapped Lasso: Stability Selection

1. **Standard Lasso Recap:**
   - Lasso (L1) shrinks coefficients, some exactly to zero → feature selection.
   - Problem: unstable if features are correlated, sample size small, or λ not perfectly tuned.
2. **Bootstrapped Lasso Idea:**
   - Resample data with replacement (bootstrap).
   - Fit Lasso on each bootstrap sample.
   - Record which features are selected.
   - Aggregate: features selected in high proportion → robustly important (stability selection).
3. **Benefits:**
   - Reduces variance in feature selection.
   - More robust to predictor correlation.
   - Provides probabilistic feature importance (e.g., selected in 85% of samples).

Post-Selection Inference:

- Statistical inference (p-values, confidence intervals, hypothesis tests) accounting for variable selection.
- Core problem:
  - Use Lasso to select variables.
  - Run OLS on selected variables → standard p-values invalid.
  - Reason: model is data-dependent (random), violating classical inference assumption that model is fixed.
- OLS assumes no correlation between X and Y outside the model. Feature selection introduces correlation → inference is biased.

Irrepresentability Condition (Sign-Consistent Variable Selection):

1. **Definition:**
   - For linear model $Y = X\beta + \epsilon$, split features into:
     - Active set S (true nonzero β_j)
     - Inactive set S^c (true zero β_j)
   - Condition:
   $$\left\| X_{S^c}^\top X_S (X_S^\top X_S)^{-1} \text{sign}(\beta_S) \right\|_\infty < 1$$
   - Intuition: inactive features must not be too correlated with active ones.
2. **Implications:**
   - Condition holds → Lasso can recover correct nonzero features (sign-consistent).
   - Condition fails → Lasso may:
     - Include irrelevant features (false positives)
     - Exclude relevant features (false negatives)
3. **Practical challenge:**
   - Real-world data often violates this condition due to collinearity.
   - Exact variable selection is not guaranteed, even if prediction performs well.

1.4 Elastic Net

The Elastic Net was proposed by Zou and Hastie (2005) to address a fundamental weakness of the Lasso in the presence of correlated predictors: among a group of highly correlated features, the Lasso tends to arbitrarily select only one and discard the rest, producing an unstable and potentially misleading variable selection. The Elastic Net combines the L_1 penalty of the Lasso (for sparsity) with the L_2 penalty of Ridge regression (for grouping correlated predictors together), via a mixing parameter α that controls the balance between the two. This hybrid penalty preserves the selection-and-shrinkage property of Lasso while inheriting the grouping effect and numerical stability of Ridge.

✓ Pros

- Combines L_1 sparsity and L_2 grouping effect; selects correlated predictors together.
- Overcomes Lasso's limitation of selecting at most n variables when $p > n$.
- More stable variable selection than pure Lasso under correlated feature regimes.
- Convex; coordinate descent solves efficiently.

✗ Cons

- Two hyperparameters (λ, α mixing ratio) increase tuning complexity; 2D grid search.
- Interpretability of mixing parameter α is not intuitive to communicate.
- Dominant regularization type depends heavily on signal structure; not self-adaptive.

▷ Technical Use Case

Preferred over pure Lasso when groups of correlated features are expected to jointly predict the outcome and group-level selection is desired. Useful when $p \gg n$ and signal may be semi-sparse. The ℓ_1/ℓ_2 mixing parameter must be treated as a structural hyperparameter informed by domain knowledge of feature correlation patterns.

★ Challenges & Advanced Considerations

- The mixing ratio α controls the balance between grouping (Ridge) and sparsity (Lasso); misspecification yields worse solutions than either pure regularizer.
- *Interview-level:* How does the elastic net penalty affect the geometry of the constraint set compared to L_1 and L_2 balls?

Elastic Net: Geometry of the Constraint Set

- **Linear regression with penalty:**

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \,\text{penalty}(\beta)$$

- **Lasso (L1):**

$$\|\beta\|_1 \leq t$$

  - Geometry (2D): diamond-shaped
  - Sharp corners encourage coefficients to hit exactly zero → sparsity
- **Ridge (L2):**

$$\|\beta\|_2 \leq t$$

  - Geometry (2D): circular or elliptical
  - Smooth boundary → shrinks coefficients continuously, rarely zero → no exact sparsity

- **Elastic Net:** convex combination of L1 and L2

$$\lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2 \leq t$$

  - Geometry (2D): "rounded diamond"
  - Retains sparsity like L1 but smoother like L2

Practical Workflow for Correlated Features and Lasso/Elastic Net:

1. Detect highly correlated features:
   - Compute correlation matrix or use Variance Inflation Factor (VIF)
2. Cluster or group strongly correlated features (e.g., correlation > 0.8)
3. Select representative features from each group:
   - Pick one feature per group based on domain knowledge or univariate importance
   - Reduces redundancy and avoids arbitrary Lasso selection within correlated groups
4. Apply Lasso or Elastic Net on the reduced, less-correlated dataset
5. Tune λ via cross-validation to select truly important features

1.5 Generalized Additive Models (GAMs)

Generalized Additive Models were introduced by Hastie and Tibshirani (1986, 1990) as a flexible extension of linear models designed to capture non-linear relationships between predictors and the response without sacrificing interpretability. The key insight was to replace the linear predictor $\sum \beta_j x_j$ with a sum of smooth, nonparametric functions $\sum f_j(x_j)$—each modeled by a spline—while retaining the additive structure that allows each feature's effect to be visualized and understood independently. GAMs were proposed as a middle ground between fully parametric linear models (too rigid) and fully nonparametric regression (too flexible and computationally intractable in high dimensions), exploiting the additive structure to avoid the curse of dimensionality.

✓ Pros

- Extends linearity assumption: $g(\mathbb{E}[Y]) = \sum_j f_j(x_j)$ captures non-linear univariate effects.
- High interpretability: each f_j is a smooth function that can be visualized independently.
- Avoids curse of dimensionality inherent to full nonparametric regression.
- Spline-based smoothers have well-understood bias-variance behavior via effective degrees of freedom.

✗ Cons

- Additivity assumption rules out interaction effects unless explicitly modeled.
- Interaction terms ($f_{jk}(x_j, x_k)$) scale poorly; tensor product splines are expensive.
- Not suitable for unstructured data or when interaction structure is unknown and complex.
- Backfitting algorithm convergence can be slow and requires careful initialization.

▷ Technical Use Case

Use when relationships are non-linear but the additive structure (no interactions or known interactions) is defensible. Particularly useful when the analyst must produce interpretable, auditable models with continuous smooth effects. Appropriate for moderate-p, moderate-n tabular settings with structural assumptions about independence of features' effects.

★ Challenges & Advanced Considerations

- Smoothing parameter selection (GCV, REML) is automatic but can over-smooth or under-smooth; monitoring effective degrees of freedom is essential.
- Tensor product interactions and random effects extensions (GAMMs) greatly increase model complexity and fitting cost.
- *Interview-level:* What is the connection between GAMs and mixed-effects models? How do you identify that the additive assumption is violated?

Generalized Additive Models (GAMs):

- Widely used in **insurance**, **healthcare**, and **finance** due to interpretability and explainability.
- Can handle non-linear relationships without manual feature engineering (e.g., polynomials).
- Interpretability + flexibility:
  - Unlike black-box models (XGBoost, deep neural nets), GAMs allow clear understanding of how each feature affects predictions.
  - Insurance: predicting claims, risk scoring
  - Healthcare: patient risk prediction, survival modeling
  - Finance: credit scoring, risk assessment
- Non-linear effects can be visualized using scatter plots or partial dependence plots to identify patterns.

1.6 Quantile Regression

Quantile regression was introduced by Koenker and Bassett (1978) as a generalization of ordinary regression that models any specified conditional quantile of the response distribution rather than exclusively the conditional mean. The motivation arose from the observation that mean regression, optimizing squared loss, is sensitive to outliers and provides only a single summary of the conditional distribution—an inadequate description when the relationship between covariates and response varies across the distribution (i.e., heteroscedasticity or asymmetric effects). By minimizing an asymmetric absolute loss (the pinball or check function), quantile regression enables analysts to characterize the full conditional distribution and is especially valuable for risk modeling, where tail behavior is the primary concern.

✓ Pros

- Models any conditional quantile $Q_\tau(Y|X)$; not limited to the conditional mean.
- Robust to outliers when estimating median ($\tau = 0.5$); uses L_1-like pinball loss.
- Does not assume homoscedasticity; captures heterogeneous distributional effects.
- No distributional assumption on errors; semiparametric.

✗ Cons

- Quantile crossing: separately fitted quantiles can violate monotonicity.
- Computationally heavier than OLS; linear program at each quantile level.
- Full conditional distributional estimation requires fitting many separate models.
- Inference (confidence intervals) requires bootstrap or asymptotic approximations.

▷ Technical Use Case

Use when interest lies in the tails or specific quantiles of the conditional distribution rather than the mean. Appropriate under heteroscedasticity, heavy-tailed errors, or when asymmetric loss

is appropriate. Useful for uncertainty quantification as an alternative to prediction intervals from mean-regression models.

★ Challenges & Advanced Considerations

- Quantile crossing requires joint estimation approaches (non-crossing quantile regression via isotonic constraints).
- Pinball loss is non-differentiable at zero; interior point methods or smooth approximations needed.
- *Interview-level:* How does quantile regression relate to Expectile regression and conformal prediction intervals?

Quantile Regression (QR) in Skewed Data:

- Unlike OLS, which looks at average impact, QR can assess effects at different parts of the distribution.
- Example: Education may have a stronger effect on low-income than high-income groups.
- Particularly useful in healthcare and insurance where data is skewed.

Insurance and Tail Risk:

- Insurance pricing is about tails, not averages. Extreme costs, not the mean, drive risk.
- QR helps understand:
  - Extreme utilization
  - Large claims
  - Cost volatility

Outliers Are Not Errors:

- In healthcare, extremely high costs are real and economically important, not mistakes.
- QR is naturally robust to skewness.
- True data errors (measurement mistakes) should be removed regardless of method.

Choosing Quantile τ:

- No universal "correct" τ; choose based on decision context.
- Examples:
  - $\tau = 0.5 \rightarrow$ median, neutral
  - $\tau = 0.9 \rightarrow$ upper tail, conservative
  - $\tau = 0.1 \rightarrow$ lower tail
- Industrial decision formula:

$$\tau \approx \frac{C_1}{C_1 + C_2}$$

  where C_1 = cost of underestimating, C_2 = cost of overestimating.

Quantile Regression vs Expectile Regression vs Conformal Prediction:

- **Quantile Regression (QR):**
  - Goal: Estimate conditional τ-th quantile of $Y \mid X$.
  - Loss function: Asymmetric absolute loss

$$L_\tau(y, \hat{y}) = \begin{cases} \tau|y - \hat{y}| & y \geq \hat{y} \\ (1-\tau)|y - \hat{y}| & y < \hat{y} \end{cases}$$

  - Robust to outliers.
- **Expectile Regression (ER):**

- Similar to QR but estimates expectiles.
- Loss function: Asymmetric squared loss

$$L_\tau(y, \hat{y}) = \begin{cases} \tau(y - \hat{y})^2 & y \geq \hat{y} \\ (1 - \tau)(y - \hat{y})^2 & y < \hat{y} \end{cases}$$

- Smooth solution, sensitive to tail values.
- Used in risk management (e.g., Expected Shortfall).

- **Conformal Prediction Intervals (CPI):**
  - Provides valid prediction intervals with finite-sample coverage guarantees.
  - Relationship to QR: Uses estimated conditional quantiles.
  - Process:
    1. Fit model (e.g., QR) to training data.
    2. Compute residuals on calibration set.
    3. Use residual quantiles to create intervals for new predictions.
  - Industrial insight: Combines robust quantile estimates with distribution-free guarantees; critical in healthcare, insurance, finance.

Classification Models

2.1 Logistic Regression

Logistic regression was developed in the 1940s–1950s by statisticians including Joseph Berkson as a principled probabilistic model for binary outcomes, addressing the inadequacy of fitting a linear model directly to a 0/1 response (which produces predictions outside $[0, 1]$ and violates the Gaussian error assumption of OLS). By applying the logistic (sigmoid) function to a linear predictor, the model maps the real line into a valid probability, enabling maximum likelihood estimation under a Bernoulli likelihood. Its elegant connection to log-odds and the exponential family made it the canonical model for binary classification and the foundation of generalized linear models (GLMs), from which it is derived as the special case with a Bernoulli family and logit link function.

✓ Pros

- Provides calibrated probability estimates under correct model specification.
- Convex log-likelihood; guaranteed global optimum via gradient-based methods.
- Coefficients directly interpretable as log-odds ratios.
- Extends naturally to multinomial (softmax) and ordinal settings.
- Well-understood regularization behavior; same L_1/L_2 properties as linear regression.

✗ Cons

- Assumes linear decision boundary in the original feature space.
- Probabilities are not calibrated when the linear assumption is violated.
- Perfect separation (complete quasi-complete separation) causes coefficient divergence and infinite MLE.
- Feature engineering burden shifts entirely to the analyst.

▷ Technical Use Case

Use when the decision boundary is approximately linear in the feature space, the output is a probability (not just a label), and interpretability of feature effects is required. The gold standard for binary classification baselines. In high-dimensional settings, L_1-regularized logistic regression is preferred for sparse signal. Convex optimization landscape makes convergence reliable.

★ Challenges & Advanced Considerations

- Complete separation is a silent problem; Firth's penalized likelihood or Bayesian logistic regression with priors resolves it.
- Probability calibration: even when AUC is high, raw probabilities may be poorly calibrated; requires isotonic regression or Platt scaling.
- *Interview-level:* Why does logistic regression output poor probabilities for imbalanced classes even with correct optimization? How does reweighting affect calibration?

2.2 Support Vector Machines (SVM)

Support Vector Machines were developed by Vapnik and Cortes (1995), building on Vapnik's earlier work on optimal hyperplane classifiers, as an approach to binary classification grounded in structural risk minimization rather than empirical risk minimization. The key insight was that among all hyperplanes that correctly classify training data, the one with the maximum margin (largest distance to the nearest training points from each class) provides the strongest generalization guarantee, as formalized by VC-dimension theory. The kernel trick, introduced to handle non-linearly separable data, was transformative: by implicitly mapping inputs to a high-dimensional (possibly infinite) feature space via a kernel function, SVMs could learn complex non-linear boundaries without ever explicitly computing the high-dimensional representation.

✓ Pros

- Maximum-margin classifier; optimal hyperplane has theoretical generalization guarantees (structural risk minimization).
- Kernel trick enables non-linear boundaries without explicit feature map computation.
- Sparse representation: only support vectors define the decision boundary; memory efficient at test time.
- Strong theoretical foundation; VC-dimension bounds generalization error.

✗ Cons

- Does not natively output calibrated probabilities; Platt scaling required.
- Training scales as $O(n^2)$ to $O(n^3)$ with standard QP solvers; impractical for large n.
- Kernel and regularization parameter (C) sensitivity requires exhaustive grid search.
- Black-box in kernel space; feature importance not directly accessible.
- Multiclass requires one-vs-one or one-vs-rest decomposition.

▷ Technical Use Case

Use when n is moderate (tens of thousands at most), the signal-to-noise ratio is high, and a non-linear boundary is required without committing to a parametric form. RBF kernel is a strong default for low-to-moderate dimensional dense feature spaces. Effective in high-

dimensional sparse spaces with linear kernel. Not appropriate when probabilistic outputs or large-scale training is required.

★ Challenges & Advanced Considerations

- Kernel selection is a structural decision, not a hyperparameter; misspecified kernels lead to systematic errors that tuning cannot fix.
- Approximate methods (SGD-based, Nyström approximation) sacrifice optimality guarantees for scalability.
- Dual formulation's reliance on inner products creates challenges with missing data; imputation must precede training.
- *Interview-level:* What is the relationship between the SVM primal and dual? When is the dual preferred? Derive the KKT conditions for the soft-margin SVM.

2.3 Decision Trees (CART)

Decision trees were formalized by Breiman, Friedman, Olshen, and Stone in the landmark CART (Classification and Regression Trees) monograph (1984), though earlier variants (ID3 by Quinlan) existed in the machine learning community. The motivation was to construct a non-parametric, interpretable model that could capture complex non-linear relationships and interactions by recursively partitioning the feature space into rectangular regions, each assigned a constant prediction. CART's appeal lay in its human-readable structure—the decision path for any prediction can be traced as a sequence of if-then rules—and its ability to handle mixed feature types without preprocessing, making it accessible to practitioners without a deep statistical background.

✓ Pros

- Fully interpretable; the decision path for any observation is traceable.
- Handles mixed feature types natively; no scaling required.
- Non-parametric; no distributional assumptions.
- Captures interactions and non-linear relationships without feature engineering.
- Fast inference: $O(\log n)$ prediction after training.

✗ Cons

- High variance; small data perturbations produce drastically different trees (instability).
- Greedy top-down splitting is not globally optimal; may miss better structures.
- Biased toward high-cardinality features in impurity-based splitting.
- Continuous variables create artificial discretization artifacts.
- Poor extrapolation: predictions are constant in leaves; cannot generalize beyond training range.

▷ Technical Use Case

Use primarily as an interpretable baseline or as a component in ensemble methods. Standalone trees are appropriate only when maximum interpretability is required and data size is small. The non-convex, combinatorial optimization landscape means the fitted tree is highly dependent on the training sample. Shallow trees (depth 2–4) are preferred for interpretability; deep trees almost always require ensembling.

★ Challenges & Advanced Considerations

- Minimum samples per leaf, max depth, and impurity criterion are strongly coupled; joint tuning required.
- Feature importance via Gini impurity is biased toward high-cardinality features; permutation importance or SHAP preferred.
- Pruning strategies (pre-pruning vs. cost-complexity pruning) have different bias-variance trade-offs.
- *Interview-level:* Prove that greedy recursive splitting cannot achieve global optimality. Under what conditions is a depth-1 tree (stump) the optimal model?

2.4 k-Nearest Neighbors (kNN)

The k-Nearest Neighbors algorithm is one of the earliest and most intuitive non-parametric methods in machine learning, with foundational theoretical analysis by Cover and Hart (1967) showing that the 1-NN classifier asymptotically achieves at most twice the Bayes error rate. Its core motivation is the instance-based learning paradigm: rather than fitting a global parametric model to the data, kNN defers all generalization to test time by finding the k training points closest to the query and aggregating their labels. This "lazy learning" approach makes no assumptions about the functional form of the decision boundary and naturally adapts to local structure in the data distribution, making it a universal approximator in the large-sample limit.

✓ Pros

- Fully non-parametric; no training phase; adapts to arbitrary decision boundaries.
- Universal approximator asymptotically (Bayes-optimal as $k \to 1$, $n \to \infty$).
- No assumptions on the data distribution; robust to unusual geometries.
- Adding new data requires no retraining.

✗ Cons

- Prediction cost is $O(n \cdot d)$; scales poorly with n and d.
- Curse of dimensionality: neighborhood concept degrades rapidly above $\sim$20 dimensions.
- Sensitive to irrelevant features and feature scaling.
- No natural handling of missing values; requires imputation.
- Memory-intensive: entire training set must be stored.

▷ Technical Use Case

Use when n is small to moderate, d is low-to-moderate, and no assumptions about the functional form are warranted. Effective when local structure in the feature space is informative and the data manifold is low-dimensional. Approximate nearest neighbor structures (KD-trees, ball trees, HNSW) extend applicability but introduce approximation error.

★ Challenges & Advanced Considerations

- Distance metric choice fundamentally determines model behavior; Euclidean distance is often inappropriate for high-dimensional or mixed-type data.
- Weighted kNN with distance weighting is superior to uniform weighting but introduces sensitivity to noise near boundaries.
- *Interview-level:* Derive the bias-variance decomposition for 1-NN. How does the Bayes

error relate to the 1-NN asymptotic error?

2.5 Naive Bayes

Naive Bayes classifiers are rooted in Bayesian probability theory and were among the first probabilistic classifiers applied to text categorization tasks in the 1960s–1990s, with foundational applied work by Mitchell and colleagues in the early machine learning era. The method applies Bayes' theorem to compute the posterior probability of each class given the observed features, but introduces the "naive" simplifying assumption that all features are conditionally independent given the class label. This assumption is almost always violated in practice, yet the resulting classifier remains surprisingly competitive—particularly for text classification with bag-of-words representations—because the decision function requires only that the rank ordering of class probabilities be correct, not that the probabilities themselves be accurate.

✓ Pros

- Extremely fast to train and predict; $O(n \cdot d)$ training.
- Scales to very high-dimensional feature spaces (e.g., text: tens of thousands of features).
- Handles missing data gracefully at inference via marginalization.
- Works well with small sample sizes due to strong inductive bias.
- Strong baseline for text classification tasks.

✗ Cons

- Conditional independence assumption is almost always violated; probability estimates are poorly calibrated.
- Cannot model feature interactions; misses correlations by construction.
- Probability outputs are unreliable; useful as a classifier, not for probability estimation.
- Zero-frequency problem without Laplace smoothing causes probability collapse.

▷ Technical Use Case

Use when feature independence is approximately satisfied or when a fast, scalable baseline is needed on high-dimensional sparse data. Gaussian NB for dense continuous features; Multinomial NB for count-based features (text); Bernoulli NB for binary features. Despite violated assumptions, often competitive in text classification due to the high dimensionality and sparsity of the feature space.

★ Challenges & Advanced Considerations

- Laplace smoothing parameter is underspecified; out-of-vocabulary terms at inference are sensitive to smoothing choice.
- When features are strongly correlated, Naive Bayes double-counts evidence; classifier may be systematically overconfident.
- *Interview-level:* Under what conditions is Naive Bayes a consistent classifier even when the independence assumption is violated? Relate to the theory of approximate Bayesian inference.

Clustering Algorithms

3.1 k-Means

The k-means algorithm was independently proposed by several researchers in the 1950s–1960s, with MacQueen's 1967 formalization and Lloyd's 1957 (published 1982) quantization algorithm being the most cited precursors. It was introduced to solve the problem of partitioning a set of observations into k groups such that within-group variation is minimized—a discrete optimization problem that is NP-hard in general but admits a simple, effective iterative heuristic: alternately assign each point to its nearest centroid (E-step) and update centroids as cluster means (M-step). The algorithm can be understood as coordinate descent on the within-cluster sum of squares objective, and equivalently as a special case of the EM algorithm for a Gaussian mixture model with equal, spherical covariances.

✓ Pros

- Simple, computationally efficient: $O(nkdI)$ per run where I is iterations.
- Scales to large datasets; mini-batch variants exist.
- Easily interpretable cluster centroids.
- Convergence guaranteed (to a local minimum) in finite iterations.

✗ Cons

- Assumes spherical, equally-sized clusters; fails on elongated, density-varying, or non-convex clusters.
- Sensitive to initialization; multiple restarts required; k-means++ mitigates but does not eliminate.
- Requires k to be specified a priori; incorrect k yields meaningless clusters.
- Non-convex optimization; converges to local optima, not global.
- Sensitive to outliers; a single outlier can dominate a centroid.

▷ Technical Use Case

Use when clusters are approximately spherical and of similar size in a dense, low-to-moderate dimensional Euclidean space. The algorithm optimizes within-cluster sum of squares (WCSS); appropriate when this objective aligns with the notion of similarity required by the task. Not appropriate for non-Euclidean data or when cluster boundaries are non-convex.

★ Challenges & Advanced Considerations

- Elbow method for selecting k is heuristic and often ambiguous; silhouette score, gap statistic, or BIC are more principled.
- Convergence to poor local optima is common; running ≥ 10 random restarts with k-means++ initialization is minimum practice.
- In high-dimensional spaces, Euclidean distance loses discriminative power; dimensionality reduction before clustering is often necessary.
- *Interview-level:* Prove that k-means is a special case of the EM algorithm for a Gaussian mixture model with equal, spherical covariances.

3.2 Gaussian Mixture Models (GMM)

Gaussian Mixture Models are a probabilistic generalization of k-means clustering, motivated by the need for a principled generative model that can represent multi-modal, elliptically shaped, or overlapping cluster structures that k-means cannot capture. A GMM posits that observations are drawn from a weighted mixture of K Gaussian distributions, each with its own mean and covariance matrix. Parameter estimation is performed via the Expectation-Maximization (EM) algorithm, which alternates between computing soft cluster responsibilities (E-step) and updating mixture parameters (M-step). The GMM framework is attractive because it yields a proper density model—enabling likelihood-based model comparison, anomaly detection via low-density regions, and principled uncertainty in assignments—and reduces to k-means as a limiting case when covariances are constrained to be equal and spherical.

✓ Pros

- Soft probabilistic cluster assignments; each point has a posterior probability per cluster.
- Handles ellipsoidal clusters of varying size, shape, and orientation.
- Principled model selection via BIC/AIC; k selection is statistically grounded.
- Generative model; can be used for density estimation and anomaly detection.

✗ Cons

- EM algorithm is susceptible to local optima; multiple restarts required.
- Covariance matrix estimation degrades in high dimensions; regularization necessary.
- Can diverge (degenerate solution) when a component collapses to a single point.
- Computationally heavier than k-means.

▷ Technical Use Case

Use when clusters have elliptical geometry, soft assignments are needed, or a generative density model is required. Prefer full covariance GMM when $n \gg d$; use diagonal or tied covariance when n is limited relative to d. BIC-optimal k provides a theoretically grounded model selection criterion absent in k-means.

★ Challenges & Advanced Considerations

- Degenerate solutions (collapsed components) require regularization of covariance matrices or minimum-variance constraints.
- Identifiability: GMM parameters are only identified up to permutation of component labels.
- In high dimensions, full covariance GMM is overparameterized; factor analysis-based GMM (e.g., MCLUST) is preferred.
- *Interview-level:* Derive the E and M steps for GMM. Under what conditions does EM converge to the global maximum?

3.3 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) was introduced by Ester, Kriegel, Sander, and Xu (1996) to address two fundamental limitations of partitional clustering methods like k-means: the inability to discover clusters of arbitrary shape and the lack of a mechanism for identifying noise points. The algorithm defines clusters as dense regions of points separated by sparser regions, using two parameters: a neighborhood radius ϵ and a minimum number of points required to form a dense region (`minPts`). Points are classified as core points (sufficient neighbors), border points (reachable from a core point), or noise (outliers), allowing DBSCAN to simultaneously discover clusters and flag anomalies—a property absent from k-means and hierarchical methods.

✓ Pros

- Discovers clusters of arbitrary shape; not limited to convex geometries.
- Explicitly identifies noise/outlier points (not forced into clusters).
- Does not require k to be specified; number of clusters is data-driven.
- Robust to outliers by design; density-based definition of cluster membership.

✗ Cons

- Requires two hyperparameters (ϵ, `minPts`) that are sensitive and domain-dependent.
- Fails when clusters have varying densities; single ϵ cannot accommodate multi-scale density.
- Scales as $O(n^2)$ without spatial indexing; $O(n \log n)$ with KD-tree or ball tree.
- Undefined behavior on high-dimensional data where density estimation degrades.

▷ Technical Use Case

Use when clusters are non-convex, of irregular shape, or when outlier identification is a primary objective. Appropriate when cluster density is approximately uniform within clusters. The ϵ-ball definition of neighborhood requires a meaningful Euclidean (or other) distance metric; less effective in high dimensions.

★ Challenges & Advanced Considerations

- Selecting ϵ: the k-nearest neighbor distance plot (sorted k-dist graph) is the standard heuristic but requires manual inspection.
- HDBSCAN extends DBSCAN to handle varying density clusters using a hierarchical approach; preferred in practice.
- Border point assignment is non-deterministic when points are reachable from multiple clusters.
- *Interview-level:* How does OPTICS generalize DBSCAN? What is the reachability distance plot and how do you extract clusters from it?

3.4 Hierarchical Clustering (Agglomerative)

Hierarchical agglomerative clustering has roots in taxonomy and numerical classification dating to the 1950s–1960s, with Ward's (1963) minimum variance linkage and the development of UPGMA in phylogenetics among the most influential formulations. The method was proposed as an alternative to flat partitional clustering to address the need for a multi-resolution view of cluster structure: rather than committing to a single value of k,

agglomerative clustering builds a complete hierarchy (dendrogram) by successively merging the two most similar clusters until all observations form a single group. This approach is especially compelling when the data has genuine hierarchical organization—biological taxonomies, document hierarchies—and when exploratory analysis requires examining structure at multiple scales of resolution.

✓ Pros

- Produces a dendrogram; cluster structure at all scales is simultaneously available.
- No pre-specification of k; cut the dendrogram at any level post-hoc.
- Deterministic (for given linkage); no random restarts required.
- Compatible with arbitrary distance matrices (non-Euclidean distances allowed).

✗ Cons

- $O(n^2)$ memory and $O(n^2 \log n)$ time; not scalable to large n.
- Linkage choice (single, complete, average, Ward) dramatically changes cluster structure; no universally optimal choice.
- Greedy bottom-up construction; early merges cannot be undone.
- Sensitive to outliers (single linkage: chaining effect).

▷ Technical Use Case

Use when n is small to moderate and dendrogram visualization or hierarchical structure interpretation is needed. Ward's linkage minimizes within-cluster variance (most similar to k-means objective); complete linkage produces compact clusters; single linkage is sensitive to chaining. Particularly useful when working with precomputed distance matrices (e.g., string edit distances, DTW for sequences).

★ Challenges & Advanced Considerations

- Cutting the dendrogram is subjective; inconsistency coefficient or the gap between consecutive merge distances can guide the cut.
- Memory constraints limit application to $n < 10{,}000$ without approximations.
- *Interview-level:* What is the ultrametric property of hierarchical clustering? How does it constrain the types of cluster structures that agglomerative methods can recover?

Dimensionality Reduction

4.1 Principal Component Analysis (PCA)

Principal Component Analysis was introduced by Pearson (1901) and independently developed by Hotelling (1933) as a method for finding the directions of maximum variance in high-dimensional data. The core motivation was dimensionality reduction: by projecting data onto a small number of orthogonal directions (principal components) that capture the greatest variance, analysts could summarize complex multivariate data in a low-dimensional space amenable to visualization and further analysis. PCA is mathematically equivalent to computing the singular value decomposition (SVD) of the data matrix, making it the optimal rank-k linear approximation of the data in terms of reconstruction error—a property that cemented its role as the foundational linear dimensionality reduction technique in statistics and machine learning.

✓ Pros

- Optimal linear compression: minimizes reconstruction error among all rank-k linear projections.
- Removes linear correlations; components are orthogonal by construction.
- Deterministic; globally optimal via SVD; no local optima.
- Well-understood variance-explained criterion for component selection.
- Computationally efficient; randomized SVD scales to large matrices.

✗ Cons

- Linear only; cannot capture non-linear manifold structure.
- Sensitive to feature scaling; requires standardization unless variables are already on comparable scales.
- Components are linear combinations of all original features; interpretability degrades with large p.
- Maximizes variance, not discriminative power; may not be optimal for downstream classification.
- Fails to preserve local structure (see t-SNE, UMAP).

▷ Technical Use Case

Use for linear dimensionality reduction, decorrelation, whitening, or as a preprocessing step before distance-based methods. Appropriate when the data manifold is approximately linear and global variance structure is the quantity of interest. Not appropriate when the goal is visualization of non-linear manifold structure or when local neighborhood structure must be preserved.

★ Challenges & Advanced Considerations

- Number of components: scree plot and explained variance ratio are heuristics; cross-validation on reconstruction error is more principled.
- Data leakage: PCA must be fit only on training data; the test set must be projected using the training-set eigenvectors.
- Probabilistic PCA (PPCA) and Factor Analysis extend PCA to handle missing data and provide a generative model.
- *Interview-level:* Derive PCA as the solution to a maximum variance optimization. What is the relationship between PCA and SVD? When is PCA equivalent to Factor Analysis?

4.2 t-SNE

t-SNE (t-distributed Stochastic Neighbor Embedding) was introduced by van der Maaten and Hinton (2008) as a non-linear dimensionality reduction method specifically designed for the visualization of high-dimensional data in two or three dimensions. It was developed in response to the failure of linear methods like PCA to reveal cluster structure in complex datasets: by modeling pairwise similarities as probabilities and minimizing the KL divergence between high- and low-dimensional similarity distributions, t-SNE can map nearby high-dimensional points to nearby positions in the embedding. The key innovation over its predecessor SNE was the use of a Student-t distribution in the low-dimensional space, which alleviates the "crowding problem"—the tendency of SNE to compress moderate distances, producing cluttered embeddings—by assigning heavier probability mass to moderate distances.

✓ Pros

- Exceptional at preserving local neighborhood structure in 2D/3D projections.
- Reveals cluster structure invisible to linear methods.
- Widely adopted as the standard high-dimensional visualization tool.

✗ Cons

- Non-convex optimization; results vary across runs due to random initialization.
- Does not preserve global structure; inter-cluster distances are not interpretable.
- Cannot project new points without re-running the full algorithm (no out-of-sample extension natively).
- $O(n^2)$ naive; Barnes-Hut approximation gives $O(n \log n)$ but perplexity sensitivity remains.
- Perplexity hyperparameter controls effective neighborhood size and strongly affects output.

▷ Technical Use Case

Use exclusively for visualization. Never use t-SNE coordinates as features for downstream modeling. Appropriate for exploratory analysis of high-dimensional data when local cluster structure is the primary interest. Must be applied after dimensionality reduction to ∼50 dims via PCA to reduce noise and computation time.

★ Challenges & Advanced Considerations

- Perplexity sensitivity: small perplexity creates many small clusters; large perplexity homogenizes structure. Multiple perplexity values should be run and compared.
- Cluster sizes and inter-cluster distances in t-SNE plots are artifacts and must not be over-interpreted.
- *Interview-level:* Compare the objective functions of t-SNE and SNE. Why does the heavy-tailed Student-t distribution in the low-dimensional space resolve the crowding problem?

4.3 UMAP

UMAP (Uniform Manifold Approximation and Projection) was introduced by McInnes, Healy, and Melville (2018) as a theoretically grounded alternative to t-SNE for non-linear dimensionality reduction, motivated by two deficiencies of t-SNE: the loss of global structure (inter-cluster distances are meaningless in t-SNE embeddings) and computational cost scaling as $O(n^2)$. UMAP is grounded in Riemannian geometry and algebraic topology: it constructs a fuzzy topological representation of the high-dimensional data manifold using weighted k-nearest neighbor graphs, then optimizes a low-dimensional embedding to have the same fuzzy topological structure. Compared to t-SNE, UMAP is faster, better preserves global topology, and supports out-of-sample projection via a parametric extension, making it the preferred embedding method for many modern workflows.

✓ Pros

- Preserves both local and global topological structure better than t-SNE.
- Significantly faster than t-SNE; scales to millions of points.
- Supports out-of-sample projection via `transform()`.
- Works on non-Euclidean metrics via custom distance functions.
- Theoretically grounded in Riemannian geometry and topological data analysis.

✗ Cons

- Non-convex optimization; stochastic; results vary across runs.
- Hyperparameters (n_neighbors, min_dist) significantly affect output; no single universally optimal configuration.
- Theoretical underpinnings are sophisticated; practical behavior can be difficult to reason about.
- Not guaranteed to faithfully represent global structure in all cases.

▷ Technical Use Case

Preferred over t-SNE for visualization when global structure matters alongside local structure, when out-of-sample extension is required, or when computational efficiency is a constraint. Also increasingly used as a preprocessing step for clustering when the manifold hypothesis is believed to hold. Applicable to non-Euclidean metric spaces.

★ Challenges & Advanced Considerations

- `n_neighbors` controls the trade-off between local and global structure; must be tuned for the specific dataset.
- `min_dist` controls compactness of embedding; small values may produce overcrowded representations.
- Using UMAP embeddings as features for downstream models requires rigorous cross-validation to prevent leakage of neighborhood structure.
- *Interview-level:* What is the fuzzy simplicial set representation in UMAP, and how does it differ from the probability distribution used in t-SNE?

4.4 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis was introduced by Fisher (1936) as a method for finding the linear combination of features that best separates two or more classes—originally applied to botanical classification problems. Fisher's criterion maximizes the ratio of between-class variance to within-class variance in the projected space, yielding a discriminant axis along which classes are most separable. The probabilistic interpretation, due to Welch and later Rao, frames LDA as the Bayes-optimal classifier under Gaussian class-conditional distributions with equal covariance matrices. LDA was proposed both as a classifier and as a supervised dimensionality reduction technique, with the key advantage that it incorporates class label information into the projection—unlike PCA, which is entirely unsupervised and may project in directions irrelevant to class separability.

✓ Pros

- Supervised dimensionality reduction; maximizes class separability in the projected space.
- Produces at most $C - 1$ discriminant axes for C classes; computationally efficient.
- Can simultaneously reduce dimensionality and perform classification.
- Closed-form solution; globally optimal under Gaussian class-conditional assumptions.

✗ Cons

- Assumes Gaussian class-conditional distributions and equal covariance matrices.
- Requires $n > p$; singular within-class scatter matrix when $p \geq n$.
- Linear discrimination only; fails for non-linear class boundaries.

- Dimensionality is bounded by $C - 1$; insufficient for fine-grained feature learning.

▷ Technical Use Case

Use when class-separability in the projected space is the primary objective and the Gaussian assumption is approximately valid. Particularly valuable as a dimensionality reduction technique before other classifiers when the number of classes C provides sufficient reduced-dimension space. Regularized LDA (RLDA) extends applicability to $p > n$ via covariance shrinkage.

★ Challenges & Advanced Considerations

- Heteroscedasticity (unequal class covariances) violates assumptions; Quadratic DA (QDA) is preferred but requires $n \gg p$ per class.
- Shrinkage estimators (Ledoit-Wolf, OAS) for covariance matrices are necessary in moderate-p settings.
- *Interview-level:* Derive the LDA criterion as a generalized eigenvalue problem. What is the relationship between LDA and Fisher's linear discriminant?

Anomaly Detection

5.1 Isolation Forest

Isolation Forest was introduced by Liu, Ting, and Zhou (2008, 2012) as an anomaly detection algorithm that takes a fundamentally different approach from density estimation or distance-based methods: rather than building a model of normal behavior and flagging deviations, it directly isolates anomalies by exploiting the observation that anomalous points are few and different, and therefore easier to isolate in a random partitioning of the feature space. The algorithm builds an ensemble of random binary trees by recursively splitting randomly selected features at random values; anomalous points, being isolated from the mass of normal data, require fewer splits to isolate and thus have shorter average path lengths in the forest. This inversion of the conventional approach—isolating anomalies rather than profiling normalcy—yields a computationally efficient, parameter-light, and scalable detector with no distributional assumptions.

✓ Pros

- Computationally efficient: $O(n \log n)$ training; linear-time scoring.
- No distributional assumptions; anomaly score based on path length in random trees.
- Scales to high-dimensional data without curse-of-dimensionality degradation.
- Requires no labeled anomaly data.

✗ Cons

- Cannot detect local anomalies embedded within dense normal regions (masking).
- Anomaly threshold selection is heuristic; contamination parameter must be estimated.
- Struggles with high-dimensional data where anomalies occupy subspaces (subspace anomalies).
- Feature interactions are not captured; anomalies defined only in full-dimensional space.

▷ Technical Use Case

Use for global anomaly detection in moderate-to-high dimensional datasets when no labeled anomalies are available. The path-length criterion is effective when anomalies are globally isolated (few and different). Extended Isolation Forest (EIF) addresses axis-parallel bias in the original formulation.

★ Challenges & Advanced Considerations

- Contamination parameter (expected proportion of anomalies) is critical; miscalibration yields poor thresholds.
- Masking effect: when anomalies cluster together, they appear normal to the forest.
- Swamping: normal points in sparse regions are incorrectly flagged.
- *Interview-level:* Why does Isolation Forest's expected path length converge to $O(\log n)$ for normal points and $O(1)$ for anomalies? What is the theoretical basis for this?

5.2 One-Class SVM (OCSVM)

The One-Class SVM was introduced by Schölkopf, Platt, Shawe-Taylor, Smola, and Williamson (1999, 2001) as an extension of the standard SVM framework to the unsupervised anomaly detection setting, where only examples of the normal class are available during training. The motivation was to leverage the kernel-based discriminative power of SVMs without requiring labeled anomaly examples. The method solves a quadratic program to find the maximum-margin hyperplane in the kernel-induced feature space that separates the training data from the origin, with the hyperparameter ν controlling the fraction of training points treated as outliers. The resulting decision function defines a compact region in input space that contains most of the normal data, and test points outside this region are flagged as anomalies.

✓ Pros

- Kernel-based; can learn complex non-linear boundaries around normal data.
- Flexible: RBF kernel captures multi-modal normal distributions.
- Strong theoretical grounding in statistical learning theory (margin maximization).

✗ Cons

- Quadratic programming training: $O(n^2)$ to $O(n^3)$; impractical for large n.
- Sensitive to hyperparameters ν and γ; tuning without labeled anomalies is difficult.
- Not robust to noisy training data; outliers in training contaminate the boundary.
- No probabilistic output; anomaly score is not directly interpretable.

▷ Technical Use Case

Use when n is small to moderate and the normal data distribution is complex and non-Gaussian. Appropriate when a kernel-defined boundary around normal data is conceptually aligned with the problem. Requires clean (anomaly-free) training data for the boundary to be meaningful.

★ Challenges & Advanced Considerations

- SGD-based One-Class SVM (sklearn implementation) approximates the objective and scales linearly but loses kernel expressiveness.
- *Interview-level:* Compare OCSVM with SVDD (Support Vector Data Description). What is the geometric difference in the boundary they define?

5.3 Local Outlier Factor (LOF)

Local Outlier Factor was introduced by Breunig, Kriegel, Ng, and Sander (2000) to address a critical limitation of global anomaly detection methods: they cannot identify local anomalies—points that are anomalous relative to their local neighborhood but not globally rare. For example, a point in a sparse region of the feature space is flagged as anomalous by global methods even if all points in that region are equally sparse and thus equally "normal" locally. LOF addresses this by computing a local reachability density for each point relative to its k nearest neighbors, then comparing this density to the densities of the neighbors: a point whose local density is much lower than its neighbors' densities receives a high LOF score and is flagged as a local outlier. This local normalization makes LOF adaptive to multi-scale density variations in the data.

✓ Pros

- Detects local anomalies relative to the density of neighboring points.
- No global distributional assumption; local density comparison is adaptive.
- Effective when normal data has multiple density modes.

✗ Cons

- $O(n^2)$ complexity without spatial indexing; slow for large n.
- Curse of dimensionality degrades local density estimation.
- Threshold selection is heuristic; LOF score is a relative ratio, not a probability.
- No out-of-sample scoring without storing the full training set.

▷ Technical Use Case

Use when anomalies are locally defined relative to their neighborhood rather than globally rare. Appropriate for datasets with non-uniform density where global methods would flag dense-cluster boundaries as anomalous. Works best in low-to-moderate dimensions.

★ Challenges & Advanced Considerations

- MinPts selection controls the scale of locality; too small yields noisy scores, too large misses local anomalies.
- LOF score interpretation: LOF ≈ 1 is normal; LOF $\gg 1$ is anomalous, but the threshold is dataset-specific.
- *Interview-level:* Derive the reachability distance and local reachability density in LOF. Why is the indirectly derived density more robust than direct density estimation?

5.4 Autoencoder-Based Anomaly Detection

Autoencoder-based anomaly detection emerged from the deep learning era as a natural application of unsupervised representation learning to anomaly detection, formalized in various forms through the 2010s. The motivation is that a neural network autoencoder—trained to compress input data into a low-dimensional bottleneck representation and then reconstruct it—will learn to efficiently reconstruct the patterns present in the training data (assumed to be normal). When presented with an anomalous input that differs structurally from the training distribution, the autoencoder fails to reconstruct it accurately, yielding a high reconstruction error that serves as the anomaly score. This approach extends naturally to high-dimensional, complex, and structured data (images, sequences, sensor streams) where classical density estimation and distance-based methods are intractable.

✓ Pros

- Learns a compressed latent representation of normal data.
- Reconstruction error serves as an anomaly score: anomalies reconstruct poorly.
- Flexible architecture; can capture complex, high-dimensional distributional structure.
- Applicable to images, time series, and tabular data.

✗ Cons

- Requires sufficient normal training data; cannot learn without examples of the normal class.
- Anomalous inputs may still reconstruct well if the encoder generalizes too broadly.
- Reconstruction threshold selection requires labeled validation anomalies or domain heuristics.
- Training stability issues for deep architectures; vanishing gradients, dead neurons.

▷ Technical Use Case

Use for high-dimensional, semi-structured, or unstructured data (images, sensor streams, text embeddings) where normal data structure is too complex for classical methods. Variational Autoencoders (VAE) provide probabilistic anomaly scores via ELBO. Most effective when the normal data occupies a low-dimensional manifold in the input space.

★ Challenges & Advanced Considerations

- Bottleneck dimension is a structural hyperparameter; too small fails to reconstruct normal data; too large fails to detect anomalies.
- Catastrophic reconstruction of anomalies: some autoencoders generalize so well they reconstruct out-of-distribution inputs, defeating the detection objective.
- *Interview-level:* When might a well-trained autoencoder fail as an anomaly detector? How does a Variational Autoencoder's ELBO-based anomaly score address some of these failure modes?

Time Series & Forecasting

6.1 ARIMA / SARIMA

ARIMA (AutoRegressive Integrated Moving Average) models were systematized by Box and Jenkins in their landmark 1970 monograph as a unified framework for the identification, estimation, and diagnostic checking of linear time series models. The methodology was proposed to provide a rigorous, data-driven approach to forecasting, in contrast to ad hoc extrapolation methods prevalent at the time. ARIMA extends stationary ARMA models to non-stationary time series via differencing (the "I" component), and the seasonal extension (SARIMA) handles periodic patterns by incorporating seasonal AR and MA terms at the period lag. The Box-Jenkins approach established a principled workflow—stationarity testing, ACF/PACF-guided order identification, maximum likelihood estimation, and residual diagnostic checking—that remains the standard protocol for classical time series analysis.

✓ Pros

- Well-founded statistical framework; diagnostic tools (ACF, PACF, Ljung-Box test) are mature.
- Prediction intervals are analytically derived; uncertainty quantification is native.
- Interpretable model structure: AR, differencing, and MA components have semantic meaning.
- Handles non-stationarity via differencing; seasonal structure via SARIMA.

✗ Cons

- Assumes stationarity after differencing; fails for structural breaks or regime changes.
- Linear model; cannot capture non-linear dynamics.
- Univariate by default (VARIMA for multivariate); does not leverage exogenous signals natively without ARIMAX.
- Manual order selection (p, d, q) requires expert judgment even with auto-ARIMA heuristics.
- Does not scale to many parallel time series without automation.

▷ Technical Use Case

Use for univariate forecasting of stationary (or stationarizable) series with moderate autocorrelation structure and when interpretable prediction intervals are required. The Box-Jenkins methodology is appropriate when n is moderate (hundreds to a few thousand observations), the series is linear, and stationarity holds after differencing.

★ Challenges & Advanced Considerations

- Unit root tests (ADF, KPSS) have low power in small samples; over- or under-differencing is common.
- Structural breaks invalidate stationarity assumptions; ARIMA models trained across regime boundaries produce biased forecasts.
- Information criteria (AIC, BIC) for order selection may produce different optimal orders; BIC penalizes complexity more harshly.
- *Interview-level:* Derive the Yule-Walker equations for AR(p). Under what conditions is the MA representation invertible?

6.2 Prophet

Prophet was introduced by Taylor and Letham at Facebook (2018) to address the practical challenges of producing reliable forecasts at scale for business time series without requiring deep expertise in classical time series modeling. Business time series—web traffic, sales volumes, engagement metrics—typically exhibit strong and multiple seasonality (daily, weekly, yearly), trend changepoints driven by business events, and irregular holiday effects that ARIMA models cannot accommodate out of the box without extensive manual configuration. Prophet decomposes the time series into an additive combination of trend (with automatic changepoint detection via a Laplace prior), seasonality (modeled via Fourier series), and user-specified holiday effects, and fits the model using a flexible Bayesian framework (Stan). The goal was to democratize forecasting by enabling analysts with domain knowledge but limited statistical expertise to produce reasonable forecasts with minimal tuning.

✓ Pros

- Additive model with explicit trend, seasonality, and holiday components; interpretable decomposition.
- Handles multiple seasonality periods (daily, weekly, yearly) natively.
- Robust to missing data and outliers in the training series.
- Automatic changepoint detection for trend shifts.

✗ Cons

- Additive decomposition assumes independent trend, seasonal, and holiday effects; multiplicative seasonality is supported but less flexible.
- Not designed for multivariate forecasting or capturing cross-series interactions.
- Uncertainty intervals rely on MCMC or Laplace approximation; may be miscalibrated.
- Performance degrades on series with complex autocorrelation structure not explained by trend/seasonality.

▷ Technical Use Case

Use for business time series with strong, regular seasonality and known holiday effects. Appropriate when the series is several months to years long, non-stationarity is driven by trend and seasonality, and analysts without deep time series expertise must interpret and modify the model. Not appropriate for high-frequency financial or sensor data.

★ Challenges & Advanced Considerations

- Changepoint prior scale controls overfitting to historical trend changes; must be tuned to avoid over-fitting trend.
- Holiday regressors must be specified manually; incomplete specification leads to systematic forecast error around events.
- *Interview-level:* How does Prophet's Bayesian changepoint model differ from a classical structural time series model (e.g., state-space framework)?

6.3 LSTM / Temporal Convolutional Networks (TCN) for Forecasting

Long Short-Term Memory networks were introduced by Hochreiter and Schmidhuber (1997) to address the vanishing gradient problem that prevented standard RNNs from learning dependencies over long time horizons. By introducing a gated memory cell architecture with input, forget, and output gates, LSTMs enable gradients to flow unchanged over many time steps, making them capable of capturing long-range temporal dependencies that are invisible to ARIMA and other classical models. Temporal Convolutional Networks (TCNs), introduced by Bai, Kolter, and Koltun (2018), were proposed as an alternative to RNNs that replaces recurrent connections with dilated causal convolutions—offering parallelizable training (unlike sequential RNNs), a controllable and explicit receptive field via dilation, and often comparable or superior performance to LSTMs on sequence modeling benchmarks.

✓ Pros

- Captures complex non-linear temporal dependencies without explicit feature engineering.
- LSTM handles variable-length sequences with learned gating mechanisms.
- TCN has receptive field controllable via dilation; computationally parallelizable.
- Supports multivariate input and multi-step ahead forecasting natively.

✗ Cons

- Requires large training datasets; poor performance on short series.
- No native uncertainty quantification without conformal prediction or MC-Dropout.
- Non-interpretable; no decomposition into trend/seasonal components.
- Vanishing gradient issues in LSTM for very long sequences; attention mechanisms partially resolve this.
- Prone to overfitting; heavy regularization (dropout, weight decay) required.

▷ Technical Use Case

Use when series are long, have complex non-linear dynamics, and multiple exogenous covariates must be integrated. Appropriate for multivariate forecasting at scale. Not appropriate for short series ($n < 1000$) or when interpretability and uncertainty quantification are primary requirements.

★ Challenges & Advanced Considerations

- Data leakage via future information in sequence construction is a common and severe error; strict temporal train-test splitting and lookback window design required.
- Teacher forcing during training vs. autoregressive inference at test time creates a distributional shift (exposure bias).
- *Interview-level:* What is the vanishing gradient problem in RNNs, and how do the LSTM gating equations specifically address it? Compare this to TCN's dilated convolution receptive field.

6.4 State Space Models / Kalman Filter

The Kalman filter was introduced by Rudolf Kalman (1960) as an optimal recursive Bayesian estimator for linear Gaussian dynamical systems, solving the problem of estimating the hidden state of a system from noisy, incomplete observations in an online, computationally efficient manner. Its development was motivated by aerospace and control engineering applications—specifically trajectory estimation for missiles and spacecraft—where the state must be updated in real time as new sensor measurements arrive. The Kalman filter provides the exact posterior distribution over the hidden state (a Gaussian, characterized by its mean and covariance) at each time step, making it both optimal and fully probabilistic. The broader state space model framework, of which the Kalman filter is the inference algorithm, has since become a foundational paradigm for modeling structured temporal dynamics in statistics, econometrics, and machine learning.

✓ Pros

- Exact Bayesian filtering for linear Gaussian state-space models.
- Online updating: new observations update the state estimate incrementally.
- Principled uncertainty quantification; posterior is Gaussian with exact covariance.
- Handles missing observations naturally via the prediction step.

✗ Cons

- Linearity and Gaussianity assumptions are frequently violated.
- Nonlinear extensions (EKF, UKF) introduce approximation error; particle filters are exact but expensive.
- Transition and observation matrices must be specified or estimated; model misspecification propagates.
- Hyperparameter estimation via EM (parameter learning) can be numerically unstable.

▷ Technical Use Case

Use for online, sequential state estimation when the system dynamics are linear Gaussian. Appropriate for sensor fusion, tracking, and structural time series decomposition. Extended Kalman Filter (EKF) handles mild non-linearity; Unscented Kalman Filter (UKF) handles stronger non-linearity without linearization. Particle filters handle arbitrary non-linear, non-Gaussian systems.

★ Challenges & Advanced Considerations

- Steady-state Kalman gain can diverge if the process noise covariance Q is misspecified.
- Covariance consistency: the filter can become overconfident (estimated covariance $<$ true covariance) due to model mismatch.
- *Interview-level:* Derive the Kalman filter predict and update equations from the joint Gaussian conditioning formula. What is the relationship to the Wiener filter?

Ensemble Methods

7.1 Random Forest

Random Forests were introduced by Breiman (2001), extending his earlier bagging (bootstrap aggregating) framework by adding randomized feature selection at each split to further decorrelate the individual trees in the ensemble. The core motivation was variance reduction without commensurate bias increase: individual decision trees have low bias but extremely high variance (small perturbations in the training data produce very different trees), and averaging many trees reduces this variance proportionally to the inverse of the number of trees, tempered by their pairwise correlation. The random feature subsampling at each split was the key innovation that reduced this inter-tree correlation below what bagging alone achieved, making Random Forest substantially more powerful than either single trees or simple bagging ensembles.

✓ Pros

- Reduces variance of decision trees via bootstrap aggregation (bagging) and random feature subsets.
- Robust to overfitting with sufficient trees; asymptotically approaches a limiting accuracy.
- Handles high-dimensional data; implicit feature selection via split frequency.
- Out-of-bag (OOB) error provides unbiased generalization estimate without a separate validation set.
- Naturally parallelizable; tree training is embarrassingly parallel.

✗ Cons

- Black-box; individual tree predictions are not directly interpretable.
- Poor extrapolation: predictions are bounded by training data range.
- Memory-intensive for large forests with deep trees.
- Biased feature importance when features have different cardinalities.
- Slower prediction than single trees or linear models.

▷ Technical Use Case

Use as the primary non-linear tabular learning baseline. Excellent bias-variance trade-off via reduction of variance without commensurate increase in bias. Performs well across a wide range of problem geometries (non-linear boundaries, interactions, mixed feature types). Requires minimal preprocessing; robust to scale and outliers.

★ Challenges & Advanced Considerations

- `max_features` controls the diversity-accuracy trade-off; $\sqrt{p}$ for classification, $p/3$ for regression are classical heuristics but should be validated.
- Feature importance via mean decrease in impurity (MDI) is biased toward high-cardinality features; permutation importance or SHAP TreeExplainer are preferred.
- Extrapolation failure: a forest's prediction outside the training range is constant (equal to the mean of the nearest leaf); fails badly on non-stationary distributions.
- *Interview-level:* Prove that bagging reduces variance by a factor of $\frac{1+(B-1)\rho}{B}$ where ρ is pairwise tree correlation. How does random feature selection reduce ρ?

7.2 Gradient Boosting (GBM / XGBoost / LightGBM / CatBoost)

Gradient Boosting was formulated by Friedman (2001) as a general framework for constructing powerful ensemble models by sequentially fitting weak learners (typically shallow decision trees) to the negative gradient of a differentiable loss function with respect to the current ensemble prediction. The key insight, developed from Breiman's earlier "arcing" ideas and Schapire and Freund's AdaBoost, was to recast boosting as gradient descent in function space: each new learner is a steepest-descent step in the space of all functions, fitted to the current pseudo-residuals. This unified view enabled boosting to be applied to any differentiable loss, far beyond AdaBoost's exponential loss. Industrial implementations—XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018)—introduced algorithmic advances (second-order optimization, histogram binning, leaf-wise growth, and ordered boosting) that made gradient boosting the dominant algorithm for tabular prediction tasks.

✓ Pros

- State-of-the-art performance on tabular data across a wide range of benchmarks.
- Bias reduction via sequential residual fitting; low-bias, low-variance with appropriate regularization.
- Flexible loss function; supports regression, classification, ranking, and custom objectives.
- XGBoost/LightGBM: highly optimized implementations with parallelism, histogram binning, and GPU support.
- CatBoost: native handling of categorical features without encoding; ordered boosting reduces target leakage.

✗ Cons

- Many hyperparameters with complex interactions; exhaustive tuning is expensive.
- Sequential training; not trivially parallelizable (within-tree parallelism only).
- Prone to overfitting with too many rounds and insufficient regularization.
- Still an ensemble; less interpretable than linear models or shallow trees.
- Sensitive to noisy labels; boosting amplifies residual noise.

▷ Technical Use Case

Default choice for tabular prediction tasks requiring maximum accuracy. Effective on both dense and sparse feature spaces. Gradient boosting reduces bias sequentially while regularization controls variance; the bias-variance trade-off is controlled through learning rate, tree depth, and number of estimators. Not appropriate for unstructured data or when real-time inference latency is critical.

★ Challenges & Advanced Considerations

- Learning rate and number of estimators are strongly coupled; early stopping on a validation set is required to avoid overfitting.
- Interaction depth (`max_depth` or `num_leaves`): deeper trees capture higher-order interactions but increase variance.
- LightGBM's leaf-wise growth can dramatically overfit on small datasets compared to XGBoost's level-wise growth.
- CatBoost's ordered boosting prevents target leakage from target statistics but may be slower.

- *Interview-level:* Derive the gradient boosting update step for a general differentiable loss. What is the connection to functional gradient descent in function space?

7.3 Stacking (Stacked Generalization)

Stacked Generalization (stacking) was introduced by Wolpert (1992) as a principled method for combining the predictions of multiple diverse base learners ("level-0 models") via a meta-learner ("level-1 model") trained on the base learners' out-of-fold predictions. The motivation was to move beyond simple ensemble averaging—which treats all base learners equally—toward a learned combination that can adaptively weight different models based on their complementary strengths. The critical insight that distinguishes stacking from naive blending is the requirement that base learners' training predictions be generated via cross-validation (out-of-fold), preventing the meta-learner from being trained on data it has already seen and thereby avoiding a subtle but severe form of data leakage.

✓ Pros

- Combines diverse base learners to reduce both bias and variance beyond any single model.
- Meta-learner can learn optimal blending weights; more flexible than simple averaging.
- Can combine models operating at different abstraction levels or from different hypothesis classes.

✗ Cons

- High data leakage risk: base learners must be trained with cross-validation to generate valid out-of-fold predictions for the meta-learner.
- Training complexity multiplies by the number of folds; computationally expensive.
- Interpretability is minimal; a stack of models is opaque.
- Overfitting risk at the meta-learner level if the held-out set is small.

▷ Technical Use Case

Use when marginal performance improvements over the best single model are critical and computational budget is not a constraint. Most effective when base learners are diverse (e.g., GBM + Random Forest + Linear Model + Neural Network). The meta-learner is typically kept simple (linear regression, logistic regression) to avoid overfitting.

★ Challenges & Advanced Considerations

- Out-of-fold prediction generation for base learners is mandatory; training base learners on full training data and then generating training meta-features is a data leakage error.
- Feature bleed: if base learner features include the target or information derived from the target, the meta-learner is contaminated.
- *Interview-level:* What is the bias-variance decomposition of a stacked ensemble? Under what conditions does stacking provably reduce generalization error compared to the best base learner?

Semi-Supervised & Self-Supervised Methods

8.1 Label Propagation / Label Spreading

Label propagation and label spreading algorithms were introduced in the early 2000s (Zhu & Ghahramani, 2002; Zhou et al., 2004) as graph-based semi-supervised learning methods designed to leverage the geometric structure of unlabeled data to improve classification when labeled examples are scarce. The core idea, rooted in the manifold assumption and cluster assumption of semi-supervised learning, is that nearby points in a similarity graph are likely to share the same label. By constructing a k-nearest-neighbor graph over all data (labeled and unlabeled) and iteratively propagating label information from labeled nodes to unlabeled nodes via graph edges—weighted by feature similarity—these methods effectively use the dense regions of the unlabeled data distribution to infer label boundaries that a purely supervised model trained only on the labeled subset could not discover.

✓ Pros

- Propagates label information through a similarity graph; leverages unlabeled data structure.
- Requires only a small set of labeled examples.
- Closed-form solution for label spreading; computationally efficient on sparse graphs.

✗ Cons

- Assumes cluster assumption: points in the same cluster share labels; violated in heterogeneous clusters.
- Graph construction (kernel, kNN radius) is critical and sensitive.
- Scales poorly with n; graph matrix is $n \times n$.
- Fails when the labeled set is not representative of the manifold.

▷ Technical Use Case

Use when labeled data is scarce but the cluster/manifold assumption is plausible: unlabeled points near labeled points are likely the same class. Appropriate for low-to-moderate n, low-dimensional (or embedded) feature spaces. Graph-based methods are most effective when the data lies on a smooth manifold.

★ Challenges & Advanced Considerations

- Label spreading's regularization parameter α controls the balance between propagated and initial labels; must be tuned.
- When the graph is disconnected, labels cannot propagate across components; connected graph construction is essential.
- *Interview-level:* Derive the label spreading update as minimizing a regularized energy function on the graph. What is the relationship to graph Laplacian regularization?

8.2 Pseudo-Labeling / Self-Training

Self-training is one of the oldest semi-supervised learning techniques, with roots tracing back to the 1960s–1970s in pattern recognition, and pseudo-labeling as its modern deep learning incarnation was popularized by Lee (2013) in the context of neural networks. The idea is straightforward: train an initial classifier on labeled data, use it to generate high-confidence "pseudo-labels" for unlabeled data, then retrain the classifier on the combined labeled and pseudo-labeled dataset. The motivation is to bootstrap the learning process by progressively

incorporating the unlabeled data in a manner consistent with the model's current beliefs, iterating until convergence. Despite its simplicity, self-training with careful confidence thresholding and modern regularization (FixMatch, FlexMatch) has proven highly effective in semi-supervised image classification benchmarks.

✓ Pros

- Simple iterative procedure; extends any supervised learner to semi-supervised setting.
- Leverages unlabeled data without requiring graph construction.
- Can dramatically improve performance when the base classifier is already reasonably accurate.

✗ Cons

- Confirmation bias: errors in pseudo-labels are amplified in subsequent iterations.
- Early iterations with low-confidence pseudo-labels introduce noise.
- Convergence is not guaranteed; may oscillate or degrade.
- Requires a confidence threshold that is difficult to calibrate without labeled validation data.

▷ Technical Use Case

Use when a reasonably accurate base model exists and a large pool of unlabeled data is available. Most effective when the decision boundary is relatively clear and pseudo-labels near the boundary are filtered using a high-confidence threshold. Widely used in computer vision and NLP fine-tuning pipelines.

★ Challenges & Advanced Considerations

- The confidence threshold is the primary lever; too low introduces label noise, too high discards most unlabeled data.
- Class imbalance in pseudo-labels can exacerbate existing imbalances in the labeled data.
- *Interview-level:* Under what theoretical conditions does self-training provably improve over the supervised baseline?

8.3 Contrastive Self-Supervised Learning (SimCLR, MoCo, BYOL)

Contrastive self-supervised learning emerged from the deep learning community around 2018–2020 as a breakthrough approach to learning powerful visual representations from unlabeled data, motivated by the enormous cost of large-scale labeled dataset annotation. The core idea is to define a pretext task—bring together (attract) two augmented views of the same image in embedding space while pushing apart (repelling) views from different images—requiring the network to learn representations invariant to semantics-preserving augmentations. SimCLR (Chen et al., 2020) demonstrated that with sufficiently strong augmentations and a non-linear projection head, contrastive pretraining matches supervised ImageNet pretraining in downstream transfer tasks. MoCo (He et al., 2020) introduced a momentum encoder and memory bank for scalable negative sampling, while BYOL (Grill et al., 2020) eliminated the need for negative pairs entirely, achieving state-of-the-art representations via an asymmetric architecture with a stop-gradient operation.

✓ Pros

- Learns rich, transferable representations from unlabeled data via augmentation-based pretext tasks.
- State-of-the-art transfer learning representations; competitive with supervised pretraining on many benchmarks.
- Flexible framework; applicable to images, audio, text, and graph data with appropriate augmentation design.
- BYOL demonstrates that negative pairs are not necessary; reduces reliance on large batch sizes.

✗ Cons

- Augmentation design is the critical modeling decision; inappropriate augmentations yield meaningless representations.
- Requires large batch sizes (SimCLR) or large memory banks/momentum encoders (MoCo) for effective negative sampling.
- Computationally expensive pretraining; impractical without significant GPU resources.
- Representation quality is evaluated empirically via downstream linear probe or fine-tuning accuracy.

▷ Technical Use Case

Use when large quantities of unlabeled data are available and label acquisition is expensive. Particularly powerful for image and audio domains. The learned representations are most valuable when fine-tuned on downstream tasks with limited labeled data. Augmentation strategy must be designed to not destroy information relevant to the downstream task.

★ Challenges & Advanced Considerations

- Representation collapse (all points map to the same embedding) is a failure mode; projector head architecture, stop-gradient, and momentum encoders are design choices that prevent this.
- Augmentation-label alignment: augmentations that are invariant to downstream task-relevant features will yield useless representations.
- *Interview-level:* Why does SimCLR require large negative batch sizes while BYOL does not? What mechanism prevents BYOL from collapsing?

Reinforcement Learning

9.1 Q-Learning / Deep Q-Networks (DQN)

Q-learning was introduced by Watkins (1989, published 1992) as a model-free, off-policy reinforcement learning algorithm that directly learns the optimal action-value function $Q^*(s, a)$—the expected cumulative discounted reward for taking action a in state s and then following the optimal policy. It was proposed to solve the fundamental RL problem without knowledge of the environment's transition dynamics, using temporal difference updates to iteratively refine the value estimates. Deep Q-Networks (DQN) were introduced by Mnih et al. at DeepMind (2013, 2015) as the first scalable application of Q-learning to high-dimensional visual state spaces (Atari games), combining Q-learning with deep convolutional neural networks and introducing two key stabilization techniques: experience

replay (to break temporal correlations between updates) and a periodically updated target network (to stabilize the bootstrapped Q-value targets).

✓ Pros

- Model-free; learns value function without knowledge of environment dynamics.
- Off-policy: can learn from experience replay buffer, improving sample efficiency.
- DQN approximates Q-function with a neural network; scales to high-dimensional state spaces.
- Well-understood convergence theory for tabular Q-learning under standard assumptions.

✗ Cons

- Overestimation bias in Q-values; addressed by Double DQN.
- Discrete action spaces only; does not extend naturally to continuous control.
- Sample inefficient in complex environments; requires millions of environment interactions.
- Catastrophic forgetting when replay buffer is too small or learning rate is too high.
- Non-stationarity of targets during training destabilizes learning.

▷ Technical Use Case

Use for discrete-action environments where the reward signal is dense enough to provide learning signal within a manageable number of steps. Appropriate when an accurate environment simulator is available (sample inefficiency is mitigated by simulation speed). Experience replay and target networks are mandatory stabilization techniques.

★ Challenges & Advanced Considerations

- Replay buffer size vs. recency trade-off: large buffers provide diverse experience but slow adaptation to non-stationary environments.
- Target network update frequency is critical; too frequent causes instability, too infrequent causes learning stagnation.
- *Interview-level:* Prove the convergence of tabular Q-learning using the contraction mapping theorem. What are the Robbins-Monro conditions on the learning rate for convergence?

9.2 Policy Gradient Methods (REINFORCE, PPO, A3C)

Policy gradient methods address a fundamental limitation of value-based RL: the difficulty of handling continuous or high-dimensional action spaces where computing the argmax over $Q(s, \cdot)$ is intractable. Williams (1992) introduced REINFORCE, the foundational policy gradient algorithm, which directly optimizes the policy $\pi_\theta(a|s)$ by computing the gradient of the expected return via the log-derivative trick (the policy gradient theorem, formalized by Sutton et al., 1999). While elegant, REINFORCE suffers from high variance gradients; actor-critic methods (A3C, Mnih et al., 2016) address this by subtracting a learned value function baseline (advantage). PPO (Schulman et al., 2017) was introduced to address training instability in trust-region policy optimization (TRPO) by replacing the complex constraint with a simple clipped surrogate objective, becoming the dominant practical policy gradient algorithm.

✓ Pros

- Directly optimizes the policy; naturally handles continuous and high-dimensional action spaces.
- PPO is stable and sample-efficient relative to vanilla policy gradients; industry standard for many RL applications.
- Actor-critic methods (A3C, A2C) reduce variance via a learned value function baseline.
- Can optimize any differentiable reward function.

✗ Cons

- High variance of policy gradient estimator; requires careful variance reduction (baselines, advantage estimation).
- Local optima; non-convex policy optimization landscape provides no global guarantees.
- Sample inefficient; on-policy methods cannot reuse experience from old policies.
- Hyperparameter sensitivity: clip ratio, entropy coefficient, learning rate must be jointly tuned.

▷ Technical Use Case

Use for continuous action spaces, complex control tasks, or when the policy must be stochastic. PPO is the recommended default for most practical RL applications due to its stability. On-policy methods require fresh environment interaction at each update; environments must be fast-to-simulate. Not appropriate for offline RL (use Conservative Q-Learning or IQL).

★ Challenges & Advanced Considerations

- The clipping threshold in PPO prevents large policy updates but must be calibrated per environment; too small prevents learning, too large causes instability.
- Reward shaping: incorrect auxiliary rewards can lead to reward hacking where the agent exploits the shaped reward rather than the true objective.
- GAE (λ) for advantage estimation introduces a bias-variance trade-off controlled by λ; $\lambda = 1$ is unbiased but high-variance; $\lambda = 0$ is biased but low-variance.
- *Interview-level:* Derive the policy gradient theorem. How does the advantage function reduce variance without introducing bias?

9.3 Model-Based RL (Dyna, MBPO, World Models)

Model-based reinforcement learning was motivated by the severe sample inefficiency of model-free methods: learning a good policy from scratch via direct environment interaction requires millions of transitions, which is intractable for physical systems where each interaction is costly or slow. The core idea, pursued since the earliest days of RL research, is to learn a model of the environment's dynamics and use it to generate synthetic experience for policy improvement without additional real interactions. Sutton's Dyna architecture (1991) introduced the principle of interleaving real experience with model-simulated experience. MBPO (Janner et al., 2019) extended this with ensemble-based uncertainty estimation to bound model error, while Ha and Schmidhuber's World Models (2018) and Hafner et al.'s Dreamer demonstrated that planning in compact latent representations learned by a world model enables sample-efficient control from high-dimensional observations.

✓ Pros

- Dramatically improves sample efficiency by planning with a learned environment model.
- Enables policy optimization without additional real-environment interaction via model rollouts.
- Applicable when real-environment interaction is expensive (robotics, scientific experimentation).

✗ Cons

- Model error compounds over long rollouts; short rollout horizons are required.
- Distribution shift between real and model-generated rollouts causes systematic bias.
- Model learning and policy optimization are coupled; errors in the model propagate to the policy.
- Complex training pipelines with multiple moving parts.

▷ Technical Use Case

Use when real-world interaction is costly or slow. Ensemble-based uncertainty estimation in the dynamics model (MBPO) allows for conservative rollouts that reduce compounding error. World Models and Dreamer extend to high-dimensional visual observations via latent-space planning.

★ Challenges & Advanced Considerations

- Model bias: systematic errors in the learned dynamics model lead to a policy that exploits the model rather than the real environment.
- Short rollout horizon in MBPO limits the benefit of model-based planning; tuning the rollout length is critical.
- *Interview-level:* What is the Dyna architecture? How does interleaving real and model-generated experience address sample efficiency?

Probabilistic & Bayesian Models

10.1 Gaussian Processes (GP)

Gaussian Processes for machine learning were developed from the statistical and kriging literature (Krige, 1951; Matérn, 1960; Sacks et al., 1989) and popularized in the ML community by Williams and Rasmussen's comprehensive monograph (2006). A GP is a distribution over functions, fully specified by a mean function and a covariance (kernel) function, providing a principled Bayesian prior over the space of possible input-output mappings. The motivation for GPs as a regression and classification tool was to produce not just point predictions but calibrated posterior distributions—quantifying uncertainty in a principled way that Bayesian theory demands. GPs became particularly central to Bayesian optimization as the surrogate model of choice, enabling sample-efficient black-box optimization by sequentially querying the function at points that balance exploration (high uncertainty) and exploitation (high expected value).

✓ Pros

- Full posterior distribution over functions; principled uncertainty quantification.
- Flexible via kernel engineering; prior beliefs about smoothness encoded explicitly.
- Exact posterior is analytically tractable for Gaussian likelihood.
- Strong performance on small datasets; optimal sample efficiency for black-box optimization.
- Used in Bayesian optimization as the surrogate model.

✗ Cons

- $O(n^3)$ training (Cholesky decomposition); $O(n^2)$ memory; impractical for $n > 10{,}000$.
- Kernel selection is a structural choice; wrong kernel yields miscalibrated uncertainty.
- Stationarity assumption of standard kernels violated in many real settings.
- Hyperparameter optimization via marginal likelihood is non-convex; multiple local optima.

▷ Technical Use Case

Use for small-to-moderate datasets where calibrated uncertainty is essential (Bayesian optimization, active learning, safety-critical systems). Sparse GP approximations (inducing points: FITC, VFE) scale to larger n at the cost of approximation. Deep kernel learning combines neural network feature extraction with GP uncertainty quantification.

★ Challenges & Advanced Considerations

- Kernel hyperparameter optimization via type-II MLE can overfit to the training data; cross-validation of predictive log-likelihood is preferred.
- Inducing point placement for sparse GPs introduces an additional variational optimization problem.
- *Interview-level:* Derive the GP posterior predictive distribution. How does the kernel function encode prior beliefs about function smoothness? What is the connection between GPs and infinite-width neural networks?

10.2 Bayesian Neural Networks (BNN)

Bayesian Neural Networks were motivated by the need to extend the Bayesian framework's principled uncertainty quantification to the expressiveness of deep neural networks. MacKay (1992) and Neal (1995) pioneered BNN approaches, observing that treating network weights as random variables rather than fixed parameters provides a posterior distribution over predictions that captures epistemic uncertainty—the uncertainty due to limited data that should decrease as more data is observed, as distinguished from aleatoric uncertainty inherent in the data generation process. The key challenge is that the posterior over weights is intractable for any non-trivial network; the field has since produced a range of approximations (variational inference, Laplace approximation, MCMC) and practical alternatives (MC-Dropout, Deep Ensembles) that achieve approximate Bayesian uncertainty quantification at varying computational cost.

✓ Pros

- Posterior over weights enables principled uncertainty decomposition (epistemic vs. aleatoric).
- Resistant to overfitting through Bayesian model averaging.
- Applicable to the full complexity of neural network architectures.

✗ Cons

- Exact posterior inference is intractable; requires MCMC, Variational Inference (VI), or Laplace approximation.
- Variational BNNs (e.g., Bayes By Backprop) introduce mean-field approximations that underestimate uncertainty.
- MCMC (HMC) is accurate but prohibitively expensive for large networks.
- Computational overhead is 2–10× vs. standard neural networks.

▷ Technical Use Case

Use when calibrated predictive uncertainty from a deep model is required. MC-Dropout and Deep Ensembles are practical approximations with competitive uncertainty quality at lower computational cost. HMC-based BNNs are appropriate only for small networks and small datasets. The mean-field assumption in VI is a strong simplification that yields underconfident posteriors.

★ Challenges & Advanced Considerations

- Mean-field VI underestimates posterior variance due to the KL divergence direction (exclusive KL); this leads to overconfident in-distribution and underconfident out-of-distribution predictions.
- Cold posterior effect: tempered likelihoods often improve performance, suggesting the standard Bayesian posterior is not the correct inductive bias for modern networks.
- *Interview-level:* Compare VI (ELBO optimization) and MCMC for posterior inference in BNNs. What does the cold posterior effect imply about the Bayesian prior in neural networks?

10.3 Hidden Markov Models (HMM)

Hidden Markov Models were introduced independently in several communities during the 1960s (Baum et al., 1966, 1970; Stratonovich) and became foundational in speech recognition through the work of Jelinek, Bahl, and Mercer at IBM in the 1970s–1980s. The HMM framework models a sequence of observations as being generated by an underlying Markov chain of discrete latent states, where each state emits an observation according to an emission distribution. The motivation was to provide a mathematically tractable generative model for sequential data with temporal structure that cannot be captured by standard i.i.d. models. The development of the Baum-Welch algorithm (a special case of EM) for parameter learning and the Viterbi algorithm for optimal state sequence decoding made HMMs computationally feasible and drove their widespread adoption across speech recognition, natural language processing, and bioinformatics.

✓ Pros

- Principled generative model for sequences with latent state structure.
- Efficient inference via the Forward-Backward algorithm ($O(T \cdot K^2)$).
- Viterbi algorithm for optimal state sequence decoding.
- Well-understood parameter learning via Baum-Welch (EM) algorithm.

✗ Cons

- Markov assumption (memorylessness) is frequently violated; long-range dependencies are not modeled.
- Fixed number of hidden states must be specified; model selection for K is non-trivial.
- Emission distribution assumption (Gaussian, Multinomial) constrains expressiveness.
- Local optima in Baum-Welch; initialization sensitive.

▷ Technical Use Case

Use for sequential data with discrete latent state transitions: speech recognition, POS tagging, biological sequence analysis, regime detection in time series. The Markov assumption makes HMM appropriate when the current observation is conditionally independent of the history given the current state.

★ Challenges & Advanced Considerations

- Label switching: HMM states are only identified up to permutation; results from multiple initializations may not correspond to the same states.
- Scaling: log-space computations are required to prevent numerical underflow in Forward-Backward.
- *Interview-level:* Derive the Forward-Backward algorithm. What is the time complexity, and how does it compare to brute-force enumeration over all state sequences?

Graph-Based Models

11.1 Graph Neural Networks (GNNs): GCN, GAT, GraphSAGE

Graph Neural Networks were developed to extend deep learning to graph-structured data, motivated by the prevalence of relational data in science and engineering—molecular structures, social networks, knowledge graphs, citation networks—where standard convolutional or recurrent architectures cannot be directly applied due to the irregular topology of graphs. Early work by Gori et al. (2005) and Scarselli et al. (2009) established the message-passing paradigm; Kipf and Welling (2017) introduced the Graph Convolutional Network (GCN) as a scalable approximation to spectral graph convolution. Hamilton et al.'s GraphSAGE (2017) added inductive capability via neighborhood sampling, and Veličković et al.'s GAT (2018) introduced attention-weighted aggregation for adaptive neighbor weighting. The unifying message-passing framework (Gilmer et al., 2017) formalized the class of all such architectures, enabling systematic analysis of their expressive power.

✓ Pros

- Natively handles graph-structured data; message passing aggregates neighborhood information.
- Inductive (GraphSAGE): can generalize to unseen nodes without retraining.
- Attention mechanisms (GAT) enable adaptive, weighted neighborhood aggregation.
- Permutation invariance is built-in by design.

✗ Cons

- Over-smoothing: deep GNNs aggregate information from too large a neighborhood; node representations become indistinguishable.
- Over-squashing: exponentially growing neighborhoods compress into fixed-size vectors; long-range information is lost.
- Scalability: full-graph training requires the entire adjacency matrix in memory; mini-batching requires neighbor sampling.
- Expressive power bounded by the Weisfeiler-Lehman graph isomorphism test.

▷ Technical Use Case

Use when data has an explicit relational graph structure: molecular property prediction, social network analysis, knowledge graph completion, recommendation systems. The graph structure must be informative; adding spurious edges degrades performance. Neighbor sampling (GraphSAGE) or cluster-based mini-batching (ClusterGCN) are required for large graphs.

★ Challenges & Advanced Considerations

- Over-smoothing bounds the effective depth; GCNII and residual connections partially address this.
- The WL-1 expressiveness ceiling means GNNs cannot distinguish certain non-isomorphic graphs; higher-order GNNs (k-WL) break this ceiling at exponential computational cost.
- *Interview-level:* Prove that GCN with L layers aggregates information from an L-hop neighborhood. What is the connection between GCN and spectral graph convolution?

Deep Learning Architectures

12.1 Feedforward Neural Networks (MLP)

The Multilayer Perceptron (MLP) traces its origins to Rosenblatt's perceptron (1958) and was given its modern multi-layer form through the development of the backpropagation algorithm, popularized by Rumelhart, Hinton, and Williams (1986). The key theoretical motivation is the Universal Approximation Theorem (Cybenko, 1989; Hornik, 1991), which established that a feedforward network with at least one hidden layer and a non-linear activation function can approximate any continuous function on a compact domain to arbitrary precision, given sufficient width. This result positioned MLPs as a general non-parametric function approximation framework, free from the parametric assumptions of linear models or the locality assumptions of kernel methods, motivating their application to any supervised learning task where the input-output mapping is complex and the functional form is unknown.

✓ Pros

- Universal approximator: any continuous function can be approximated to arbitrary precision.
- Highly flexible; applicable to any fixed-dimensional input/output.
- Benefits from GPU parallelism; scales to large datasets with SGD.
- Rich ecosystem of regularization techniques (dropout, batch norm, weight decay).

✗ Cons

- No inductive bias for structured data (images, sequences, graphs); requires large datasets to learn structure.
- Non-convex optimization landscape; convergence to local optima; no global guarantees.
- Sensitive to initialization, learning rate, and architecture design.
- Opaque; interpretability requires post-hoc methods.

▷ Technical Use Case

Use for fixed-dimensional tabular or embedding inputs where no explicit structural prior exists. Appropriate when sufficient labeled data is available and the relationship is non-linear and high-order. MLPs are outperformed by gradient boosting on tabular data in most benchmarks; prefer gradient boosting as the default unless data scale ($n > 10^6$) favors neural networks.

★ Challenges & Advanced Considerations

- Dead neurons (ReLU saturation): neurons that always output 0 after initialization; leaky ReLU, ELU, or careful initialization (He initialization) mitigates.
- Batch normalization introduces a train-test discrepancy via running statistics; incorrect use during inference is a common implementation bug.
- *Interview-level:* What is the lottery ticket hypothesis? What does it imply about the role of overparameterization in neural network training?

12.2 Convolutional Neural Networks (CNN)

Convolutional Neural Networks were introduced to exploit the spatial structure of image data through the principles of local connectivity, weight sharing, and translation equivariance—inspired by the hierarchical processing structure of the mammalian visual cortex (Hubel and Wiesel, 1962). LeCun et al.'s LeNet (1989, 1998) demonstrated that convolutional layers, combined with pooling and fully connected layers trained end-to-end via backpropagation, could achieve state-of-the-art handwritten digit recognition. The critical insight was that visual patterns (edges, textures, shapes) are locally structured and translationally invariant: the same filter detecting a horizontal edge should apply everywhere in the image. This inductive bias reduces the parameter count dramatically compared to a fully connected network, enabling effective training from relatively limited labeled data—a property that became transformative after Krizhevsky et al.'s AlexNet won the 2012 ImageNet competition by a large margin.

✓ Pros

- Translation equivariance and weight sharing provide powerful inductive bias for spatial data.
- Dramatically reduces parameter count compared to fully connected alternatives.
- Hierarchical feature learning: low-level edges to high-level semantic features.
- State-of-the-art for image classification, object detection, and segmentation.

✗ Cons

- Translation equivariance but not rotation or scale invariance (without augmentation or specialized architectures).

- Requires substantial labeled data; transfer learning from large pretrained models is typically necessary.
- Fixed receptive field per layer; very deep networks required to aggregate global context.
- Black box; feature maps require visualization tools for interpretation.

▷ Technical Use Case

Use for image, video, and 1D signal data where local feature patterns are translationally invariant. Transfer learning from ImageNet pretrained models (ResNet, EfficientNet, ViT) is the standard approach when labeled data is limited. Vision Transformers (ViT) challenge CNN dominance for large-scale image tasks; CNNs remain competitive in data-limited regimes.

★ Challenges & Advanced Considerations

- Stride and padding choices affect spatial resolution in ways that interact with batch normalization and pooling; architectural design requires careful tracking.
- Feature pyramid networks and multi-scale architectures address the scale-sensitivity limitation for detection tasks.
- *Interview-level:* Derive the computational complexity of a convolutional layer in terms of input size, kernel size, and number of filters. Compare to a fully connected layer.

12.3 Transformers (Attention Mechanism)

The Transformer architecture was introduced by Vaswani et al. (2017) in the paper "Attention Is All You Need" as a replacement for recurrent neural networks in sequence-to-sequence tasks, motivated by two fundamental limitations of RNNs: the sequential computation that prevents parallelization during training, and the difficulty of learning dependencies between positions far apart in the sequence due to vanishing gradients. The self-attention mechanism allows every position in the sequence to directly attend to every other position in a single layer, collapsing long-range dependencies to $O(1)$ path length regardless of sequence length. Combined with positional encodings and multi-head attention, the Transformer became the foundation of modern NLP (BERT, GPT), computer vision (ViT), and multimodal learning, fundamentally restructuring deep learning around the attention primitive.

✓ Pros

- Global context aggregation via self-attention; no locality assumption.
- Highly parallelizable during training (unlike RNNs); scales with compute.
- Foundation of state-of-the-art architectures across NLP, vision, and multimodal domains.
- Flexible: encoder-only (BERT), decoder-only (GPT), encoder-decoder (T5) variants cover a broad range of tasks.
- Positional encoding is explicit and modifiable; not constrained to sequence ordering.

✗ Cons

- Attention is $O(n^2)$ in sequence length; quadratic complexity is a fundamental bottleneck for very long sequences.
- Requires massive datasets and compute for pretraining from scratch; fine-tuning a pretrained model is the practical paradigm.
- No inductive bias for locality; underperforms CNNs on image tasks with limited data.

- Positional encoding choice can significantly affect performance on long sequences.

▷ Technical Use Case

Use for sequence-to-sequence, generative language modeling, or any task where global context integration is essential. Always start from a pretrained foundation model (BERT, GPT, T5, LLaMA) and fine-tune rather than training from scratch. Linear attention variants (Performer, FlashAttention) address the quadratic complexity for long contexts.

★ Challenges & Advanced Considerations

- Fine-tuning all layers vs. parameter-efficient fine-tuning (LoRA, Adapter layers, Prefix tuning) is a critical design decision; full fine-tuning risks catastrophic forgetting and is compute-expensive.
- Attention head interpretability is limited; specific heads develop interpretable roles but aggregate behavior is complex.
- *Interview-level:* Derive the scaled dot-product attention formula. Why is the scaling factor $\frac{1}{\sqrt{d_k}}$ necessary? What is multi-head attention achieving computationally compared to single-head attention?

Optimization Methods in ML

13.1 Stochastic Gradient Descent (SGD) and Variants

Stochastic Gradient Descent has its roots in the Robbins-Monro stochastic approximation algorithm (1951), introduced as a method for finding the root of a regression function from noisy observations. In the context of machine learning, SGD was motivated by the computational intractability of computing the full gradient of the loss over the entire training dataset at each optimization step as dataset sizes grew to millions of examples. By estimating the gradient from a random mini-batch of examples, SGD makes rapid, noisy gradient steps that, despite their noise, converge to a minimum in expectation. The noise introduced by mini-batch sampling acts as an implicit regularizer—helping escape sharp local minima and find flatter, more generalizable solutions—a property that distinguishes SGD's generalization behavior from full-batch gradient descent.

✓ Pros

- Noisy gradient estimates from mini-batches regularize training; beneficial for generalization (implicit regularization).
- Scales to arbitrary dataset sizes via mini-batch processing.
- Momentum, Nesterov momentum, and learning rate schedules dramatically improve convergence.
- SGD with carefully tuned learning rate and momentum outperforms adaptive methods on image classification generalization.

✗ Cons

- Highly sensitive to learning rate; manual tuning or warm-up schedules required.
- Slow convergence on ill-conditioned loss landscapes; uniform learning rate across all parameters is suboptimal.
- Requires careful weight initialization to avoid vanishing/exploding gradients.

▷ Technical Use Case

Use SGD with momentum and learning rate scheduling for large-scale image classification and vision tasks where generalization of the converged model is prioritized over convergence speed. Adam is preferred for rapid convergence in NLP and fine-tuning. The choice between SGD and adaptive methods is a bias-variance trade-off in the optimization trajectory.

★ Challenges & Advanced Considerations

- Learning rate warm-up is essential for Transformers (Vaswani et al. schedule); without it, early iterations can destabilize layer normalization.
- Batch size scaling: increasing batch size requires proportional learning rate scaling (linear scaling rule) to maintain convergence behavior.
- *Interview-level:* Why does SGD with small batch sizes often generalize better than large-batch SGD, despite both converging to similar training loss values?

13.2 Adam / AdamW

Adam (Adaptive Moment Estimation) was introduced by Kingma and Ba (2015) to address the sensitivity of SGD to the learning rate and its uniform treatment of all parameters. SGD applies the same learning rate to all parameters, which is suboptimal when gradient magnitudes vary dramatically across parameters or when gradients are sparse (as in NLP with word embeddings). Adam maintains exponential moving averages of both the first moment (mean) and second moment (uncentered variance) of the gradients, using them to compute adaptive per-parameter learning rates. AdamW (Loshchilov & Hutter, 2019) was introduced as a correction to Adam's improper implementation of L_2 regularization: in Adam, weight decay applied to the gradient update is coupled with the adaptive learning rate, making the regularization non-uniform; AdamW decouples weight decay from the gradient update, restoring the intended regularization effect.

✓ Pros

- Adaptive per-parameter learning rates; handles sparse gradients and ill-conditioned loss surfaces efficiently.
- Fast convergence; default optimizer for most NLP and fine-tuning tasks.
- AdamW correctly decouples weight decay from gradient adaptation (resolves Adam's L_2 regularization inconsistency).

✗ Cons

- Generalization gap: Adam converges to flatter local minima less reliably than SGD in some vision tasks.
- Memory overhead: maintains first and second moment estimates per parameter (2× parameter memory).
- $\beta_1, \beta_2, \epsilon$ are additional hyperparameters (though defaults are generally stable).

▷ Technical Use Case

Default optimizer for Transformers, RNNs, and NLP tasks. AdamW is preferred over Adam when L_2 regularization is used. The adaptive step size makes Adam robust to gradient scale differences across layers, which is particularly important in deep networks with batch normalization.

★ Challenges & Advanced Considerations

- Weight decay in Adam is applied to the gradient update, not the weight directly; this conflates regularization with gradient adaptation. AdamW separates these correctly.
- Adam can diverge with too-large ϵ or too-small β_2; default values are stable across most tasks.
- *Interview-level:* Derive the Adam update equations. Why does the bias correction in the first and second moments matter in early training iterations?

13.3 Second-Order Methods (L-BFGS, Natural Gradient)

Second-order optimization methods were developed to accelerate convergence beyond what first-order (gradient-only) methods can achieve, motivated by the observation that the gradient alone does not encode curvature information about the loss landscape. Newton's method uses the full Hessian to scale and rotate the gradient step, achieving quadratic convergence near optima—but computing and inverting the Hessian is $O(p^3)$ in the number of parameters. L-BFGS (Nocedal, 1980; Liu & Nocedal, 1989) approximates the inverse Hessian using a limited-memory rank-$2m$ update from the last m gradient differences, achieving superlinear convergence at $O(mp)$ cost per step. The natural gradient (Amari, 1998) provides an alternative second-order method that scales the gradient by the inverse Fisher information matrix—the Riemannian metric of the statistical manifold—making optimization invariant to reparametrization of the model.

✓ Pros

- L-BFGS approximates the inverse Hessian; superlinear convergence near optima.
- Natural gradient accounts for the Fisher information matrix; optimization is invariant to reparametrization.
- Deterministic convergence guarantees for convex problems; far fewer iterations required.

✗ Cons

- L-BFGS requires full-batch gradients; impractical for large n.
- Fisher matrix computation and inversion is $O(p^2)$ to $O(p^3)$; intractable for deep networks.
- Natural gradient requires Kronecker-factored approximations (KFAC) to be computationally feasible.
- Does not generalize well to stochastic settings without modification.

▷ Technical Use Case

Use L-BFGS for small-to-moderate scale problems (shallow models, tabular data) where full-batch gradient computation is feasible and fast convergence is required. Natural gradient (via KFAC) is used in research contexts for Kronecker-factored optimization of deep networks. Convex problems benefit most from second-order methods.

★ Challenges & Advanced Considerations

- L-BFGS memory parameter m (number of vector pairs stored) controls the quality of the Hessian approximation vs. memory trade-off.
- KFAC introduces block-diagonal approximations of the Fisher matrix; the factored approximation quality depends on layer structure.
- *Interview-level:* Derive the natural gradient update. Why is it invariant to reparametriza-

tion of the model's parameters, and why does this matter for optimization?

Model Interpretation & Explainability

14.1 SHAP (SHapley Additive exPlanations)

SHAP was introduced by Lundberg and Lee (2017) as a unified framework for feature attribution in machine learning, grounded in cooperative game theory. The motivation was the proliferation of ad hoc and inconsistent feature importance methods—permutation importance, LIME weights, gradient-based attributions—each satisfying different, incompatible desiderata, making it impossible to compare or trust attributions across methods. By identifying model explanations with Shapley values from game theory, SHAP provides a unique attribution that satisfies four axiomatic properties simultaneously: efficiency (attributions sum to the prediction), symmetry, dummy (irrelevant features get zero attribution), and additivity. The TreeSHAP algorithm (Lundberg et al., 2018) computes exact Shapley values for tree-based models in polynomial time, making rigorous attribution practical for the most commonly used ML models.

✓ Pros

- Unique: only feature attribution method satisfying all four Shapley axioms (efficiency, symmetry, dummy, additivity).
- Locally consistent and model-agnostic; applicable to any model.
- TreeSHAP provides exact, polynomial-time computation for tree-based models.
- Global importance can be derived by aggregating local explanations (mean absolute SHAP).

✗ Cons

- Exponential exact computation for general models; kernel SHAP is an approximation.
- Assumes feature independence when computing marginal contributions; misleading under strong correlations.
- Explains the model, not the true data-generating process; can confirm model bias rather than reveal truth.
- SHAP values for correlated features distribute importance in ways that can be misleading.

▷ Technical Use Case

Use as the primary feature attribution method for tree-based models (TreeSHAP) and as a principled local explainability framework for model-agnostic settings. Mandatory for high-stakes model auditing. SHAP interaction values capture pairwise interactions; DeepSHAP handles neural networks but requires specific architectural assumptions.

★ Challenges & Advanced Considerations

- Under feature correlation, SHAP distributes credit among correlated features in ways that may not correspond to causal attribution; conditional SHAP mitigates but is computationally expensive.
- SHAP explains model behavior, not causal effects; a high SHAP value for a feature does not imply causality.
- *Interview-level:* Prove that SHAP values are the unique attribution satisfying the four

Shapley axioms. What is the difference between marginal and conditional SHAP, and when does each produce misleading explanations?

14.2 LIME (Local Interpretable Model-agnostic Explanations)

LIME was introduced by Ribeiro, Singh, and Guestrin (2016) as a practical, model-agnostic method for explaining individual predictions of any black-box classifier or regressor. The motivation was the growing deployment of opaque models (complex ensembles, neural networks) in settings where human oversight and explanation of individual decisions are necessary—for instance, explaining why a specific loan application was rejected. The key insight is that while a model's global decision boundary may be complex and non-linear, its behavior in the neighborhood of any individual prediction can often be approximated by a simple, interpretable linear model. LIME generates explanations by perturbing the input, obtaining the black-box model's predictions on the perturbed samples, and fitting a weighted sparse linear model (with proximity to the original point as weights) to explain the local decision boundary.

✓ Pros

- Explains any black-box model locally via a sparse linear approximation.
- Intuitive explanations expressed as feature weights of a locally fitted linear model.
- Model-agnostic; works with any classifier or regressor.

✗ Cons

- Explanations are unstable: small changes to the query point or sampling can produce very different explanations.
- Locality kernel bandwidth (σ) is a critical hyperparameter with no principled selection method.
- Perturbation distribution for tabular data ignores feature correlations.
- Linear local approximation may be a poor fit for highly non-linear decision boundaries.

▷ Technical Use Case

Use for quick, human-readable local explanations of individual predictions. LIME is less principled than SHAP (does not satisfy Shapley axioms) but is faster and more accessible. Prefer SHAP for rigorous attribution; use LIME for rapid exploration or when SHAP computation is prohibitive.

★ Challenges & Advanced Considerations

- Instability of explanations is a fundamental limitation; multiple runs with the same query point can produce different explanations due to stochastic sampling.
- For text and image models, the superpixel/word segment definition is a structural choice that affects explanation quality.
- *Interview-level:* How does LIME define the local neighborhood? Why is the explanation instability a structural problem rather than an implementation bug?

14.3 Partial Dependence Plots (PDP) & Individual Conditional Expectation (ICE)

Partial Dependence Plots were introduced by Friedman (2001) in the context of gradient boosting machines as a diagnostic tool for understanding the marginal effect of one or two features on the model's predictions, marginalized over the joint distribution of all other features. The motivation was to make complex ensemble models interpretable by decomposing the joint prediction function into per-feature effect plots, analogous to the effect plots of GAMs but applicable to any model. Individual Conditional Expectation (ICE) plots were introduced by Goldstein et al. (2015) to reveal the heterogeneity in feature effects that PDPs obscure by averaging: where a PDP shows the average effect of a feature, ICE plots show one line per observation, revealing whether the effect of the feature is qualitatively different for different subpopulations—a signal of interaction effects that PDPs cannot detect.

✓ Pros

- PDP: visualizes the marginal effect of one or two features on model predictions.
- ICE: shows per-instance variation, revealing heterogeneous effects masked by the PDP average.
- Model-agnostic; applicable to any prediction model.
- Intuitive visualization of non-linear feature effects.

✗ Cons

- PDP marginalizes over all other features; misleading when features are correlated (extrapolation into unobserved feature combinations).
- Computationally expensive for large n; requires model evaluation over a grid.
- Two-way PDPs are the practical limit; higher-order interactions cannot be visualized.

▷ Technical Use Case

Use for global model debugging and understanding the functional form of feature effects. ICE plots are essential when effect heterogeneity is expected. Accumulated Local Effects (ALE) plots should be preferred over PDP when features are correlated, as ALE conditions on feature values rather than marginalizing over the joint distribution.

★ Challenges & Advanced Considerations

- PDP under feature correlation: averaging over the marginal distribution of other features creates impossible feature combinations; ALE plots address this by computing local averages along the feature axis.
- Centered ICE (c-ICE) normalizes curves to a reference point; reveals crossing patterns that indicate interaction effects.
- *Interview-level:* Derive the mathematical definition of ALE and explain why it is preferable to PDP in the presence of feature correlation.

14.4 Permutation Feature Importance

Permutation feature importance was introduced by Breiman (2001) as part of the Random Forest framework, motivated by the need for a feature importance measure that reflects a feature's contribution to the model's predictive performance rather than to its training-time fitting criterion. The idea is conceptually elegant: randomly permuting the values of a feature in the held-out test set breaks the association between that feature and the target while leaving all other features intact; the resulting drop in model performance (measured by any chosen metric) quantifies how much the model relied on that feature for accurate predictions. This approach is model-agnostic, metric-flexible, and evaluated on held-out data—addressing the bias of in-training impurity-based importance toward high-cardinality features.

✓ Pros

- Model-agnostic; applicable to any model using any performance metric.
- Directly measures the contribution of each feature to model performance on held-out data.
- Detects features that improve training metrics but not generalization.

✗ Cons

- Permutation breaks correlations; feature importance of correlated features is distributed arbitrarily.
- High variance; multiple permutations per feature required.
- Permutation of test data only; result depends on the test set distribution.
- Can underestimate importance of features with weak marginal effects but strong interactions.

▷ Technical Use Case

Use as a model-agnostic global importance measure. Prefer over Gini-impurity-based importance for tree models due to its bias correction and applicability to any metric. Must be computed on held-out data to measure generalization importance rather than training-set fit.

★ Challenges & Advanced Considerations

- Under feature correlation, permuted importance distributes importance between correlated features arbitrarily; removing one correlated feature allows the other to compensate.
- Number of permutations: variance of the estimate decreases with more permutations; 10–50 repetitions are standard.
- *Interview-level:* How does permutation importance differ from SHAP? In what situations would you prefer each?

Causal Inference Methods

15.1 Propensity Score Methods (Matching, Weighting)

Propensity score methods were introduced by Rosenbaum and Rubin (1983) as a practical solution to the problem of controlling for confounding in observational studies when the number of covariates is large. In experimental settings, randomization ensures that treatment and control groups are balanced on all covariates (observed and unobserved); in observational

studies, no such balance exists. Directly matching on all covariates simultaneously is infeasible when the covariate space is high-dimensional. Rosenbaum and Rubin proved the propensity score theorem: conditioning on a single scalar—the probability of treatment given observed covariates—is sufficient to remove confounding by all observed covariates. This dimension-reduction result made covariate adjustment tractable, enabling a range of estimators (matching, stratification, inverse probability weighting) that balance covariate distributions between treatment groups without parametric outcome modeling assumptions.

✓ Pros

- Reduces confounding bias by balancing covariates between treatment and control groups.
- Propensity score theorem: balancing on $e(X) = P(T = 1|X)$ is sufficient for covariate balance.
- Flexible: matching, stratification, IPTW, and doubly robust estimators.
- Does not require a structural equation model; observational study feasible.

✗ Cons

- Requires strong ignorability (no unmeasured confounders); this assumption is untestable.
- Positivity assumption (overlap): every unit must have positive probability of receiving either treatment; violation leads to extreme weights.
- Propensity score model misspecification introduces bias.
- Matching discards units; reduces effective sample size.

▷ Technical Use Case

Use for estimating average treatment effects (ATE, ATT) from observational data when randomization is infeasible. Doubly robust estimators (combining outcome model and propensity model) provide protection against misspecification of one of the two models. The key identifying assumption (no unmeasured confounding) must be defended on substantive grounds.

★ Challenges & Advanced Considerations

- Extreme IPTW weights indicate positivity violations; trimming or truncating weights introduces bias-variance trade-off.
- Doubly robust estimators (AIPW) are consistent if either the outcome model or the propensity model is correctly specified, but not both misspecified.
- *Interview-level:* Prove the doubly robust property of the AIPW estimator. When does IPTW achieve the semiparametric efficiency bound?

15.2 Instrumental Variables (IV)

Instrumental Variables estimation has roots in econometrics dating to Working (1927) and Wright (1928), developed to address the endogeneity problem: when the treatment variable is correlated with unmeasured confounders (omitted variables, simultaneous causation, measurement error), OLS estimates of causal effects are biased and inconsistent. The IV approach exploits an exogenous instrument—a variable that is correlated with the treatment but affects the outcome only through the treatment—to isolate variation in the treatment that is free of confounding. Two-Stage Least Squares (2SLS) became the dominant implementation, with the first stage regressing treatment on the instrument and the second

stage regressing the outcome on the predicted treatment. Angrist, Imbens, and Rubin (1996) provided the modern LATE interpretation, establishing what causal quantity IV identifies in the presence of non-compliance (the local average treatment effect for compliers).

✓ Pros

- Identifies causal effects in the presence of unmeasured confounders via an exogenous instrument.
- Consistent estimation of LATE (Local Average Treatment Effect) under weak monotonicity assumptions.
- 2SLS and related IV estimators are well-studied with robust inference procedures.

✗ Cons

- Valid instrument is extremely difficult to find; instrument validity assumptions are untestable.
- Weak instruments lead to highly variable estimates; first-stage F-statistic < 10 is a warning sign.
- Estimates LATE (complier population), not ATE; external validity is limited.
- Exclusion restriction (instrument affects outcome only through the treatment) is a strong, untestable assumption.

▷ Technical Use Case

Use when unmeasured confounders are present and a valid instrument can be credibly argued. The instrument must satisfy: relevance (strong correlation with treatment), exogeneity (uncorrelated with unmeasured confounders), and exclusion restriction (affects outcome only through treatment). Natural experiments (policy changes, random assignment) provide the most credible instruments.

★ Challenges & Advanced Considerations

- Weak instrument bias: with weak instruments, 2SLS is inconsistent and biased toward OLS; LIML is more robust to weak instruments.
- Multiple instruments require overidentification tests (Sargan-Hansen) to detect instrument invalidity.
- *Interview-level:* Derive the 2SLS estimator as a two-stage procedure. What is the LATE theorem, and what population does it identify?

15.3 Difference-in-Differences (DiD)

Difference-in-Differences is a quasi-experimental estimation strategy that has been used in economics and social sciences since at least Snow's (1855) cholera study, formalized in modern econometrics through Card and Krueger's (1994) influential minimum wage study. The method estimates a causal treatment effect by comparing the change in outcomes over time for a treated group versus a control group, differencing out time-invariant confounders (group fixed effects) and common time trends (time fixed effects) simultaneously. The identifying assumption—parallel trends: in the absence of treatment, the treated and control groups would have followed parallel outcome trajectories—makes DiD applicable whenever a credible control group exists, even when the treatment is not randomly assigned, making it the workhorse of modern program evaluation and policy analysis.

✓ Pros

- Intuitive and widely used; controls for time-invariant unit-level confounders.
- Parallel trends assumption is partially testable via pre-treatment trends.
- Generalizes to staggered adoption designs (though requires care).
- Applicable to panel data settings without an instrument.

✗ Cons

- Parallel trends assumption is the critical untestable identifying assumption.
- Staggered treatment adoption with heterogeneous treatment effects: naive 2×2 DiD is biased; Callaway-Sant'Anna or Borusyak et al. estimators required.
- SUTVA (no interference between units) must hold; violated in network settings.

▷ Technical Use Case

Use for policy evaluation and quasi-experimental designs with panel data when the treatment is assigned at the group level and the parallel trends assumption is defensible. Event study plots of pre-treatment coefficients are the standard diagnostic for parallel trends plausibility.

★ Challenges & Advanced Considerations

- Heterogeneous treatment effects in staggered DiD: late-adopters serve as controls for early-adopters in the canonical 2×2 decomposition, potentially weighting by negative effects; modern estimators explicitly handle this.
- Inference: standard errors must account for serial correlation within units (cluster-robust SE) and potential spatial correlation.
- *Interview-level:* Derive the two-way fixed effects (TWFE) estimator for DiD. Why does TWFE fail with staggered adoption and heterogeneous effects?

15.4 Regression Discontinuity Design (RDD)

Regression Discontinuity Design was introduced by Thistlethwaite and Campbell (1960) and rigorously formalized in modern econometrics by Hahn, Todd, and van der Klaauw (2001) and Imbens and Lemieux (2008). RDD exploits the fact that in many real-world settings, treatment is assigned based on whether a continuous "running variable" (test score, income, age) crosses a known threshold—creating a sharp discontinuity in treatment probability at the cutoff. Because units just above and just below the threshold are essentially identical in all other respects (they differ only in which side of the cutoff they fall), the discontinuity in outcomes at the threshold identifies the causal treatment effect for units near the cutoff without randomization. RDD was proposed as one of the most credible quasi-experimental designs precisely because the identification assumption (continuity of potential outcomes at the cutoff) is largely testable and transparent.

✓ Pros

- Exploits sharp or fuzzy thresholds in assignment rules for near-experimental identification.
- Local average treatment effect is estimated at the cutoff; does not require global functional form.
- Transparent and visually verifiable; continuity of covariates at the cutoff tests validity.

✗ Cons

- External validity: identifies treatment effect only at the cutoff; may not generalize away from it.
- Bandwidth selection around the cutoff introduces a bias-variance trade-off.
- Manipulation of the running variable (sorting at the cutoff) invalidates the design; McCrary density test required.
- Requires sufficient observations near the cutoff; otherwise underpowered.

▷ Technical Use Case

Use when treatment assignment is determined by a continuous running variable crossing a known threshold. Sharp RDD requires all units above the threshold to be treated; fuzzy RDD (instrumental variable approach) handles partial compliance. Nonparametric local polynomial estimation is preferred over parametric global functional forms.

★ Challenges & Advanced Considerations

- Bandwidth selection: MSE-optimal (IK bandwidth) vs. CER-optimal (coverage error rate) bandwidths are available; Calonico-Cattaneo-Titiunik (CCT) robust bias-corrected inference is the current standard.
- Continuity of baseline covariates at the cutoff tests the assumption that units close to the cutoff are exchangeable.
- *Interview-level:* What is the continuity assumption in RDD, and how does the McCrary density test operationalize a test of its plausibility?

Generative Models

16.1 Variational Autoencoders (VAE)

Variational Autoencoders were introduced by Kingma and Welling (2014) and independently by Rezende, Mohamed, and Wierstra (2014) as a method for learning deep generative models with tractable inference, addressing the intractability of exact posterior inference in deep latent variable models. The core idea is to simultaneously learn a generative model $p_\theta(x|z)$ (decoder) and an approximate posterior $q_\phi(z|x)$ (encoder) by maximizing the Evidence Lower BOund (ELBO) on the log-likelihood, made differentiable through the reparameterization trick. VAEs were proposed as a scalable alternative to MCMC-based Bayesian deep learning: by amortizing inference across all data points through a shared encoder network, VAEs enable efficient approximate posterior inference and structured latent space learning—properties that made them the foundation of probabilistic generative modeling in the deep learning era.

✓ Pros

- Principled probabilistic framework; learns a smooth, structured latent space.
- Evidence lower bound (ELBO) provides a tractable objective combining reconstruction and regularization.
- Enables latent space interpolation and sampling.
- Disentangled representations possible via β-VAE variants.

✗ Cons

- Generated samples are blurry for image data; the Gaussian decoder's pixel-wise MSE loss averages over modes.
- Posterior collapse: the decoder ignores the latent code when the model over-regularizes via KL.
- Mean-field approximation of the posterior underestimates true posterior complexity.
- The evidence lower bound does not tightly bound the true marginal likelihood.

▷ Technical Use Case

Use for structured generative modeling, anomaly detection via reconstruction error + KL divergence, latent space exploration, and representation learning. VAE is preferred over GAN when training stability and mode coverage are prioritized over sharpness of generated samples. Conditional VAE (CVAE) enables class-conditional generation.

★ Challenges & Advanced Considerations

- Posterior collapse prevention: KL annealing, free bits, and skip connections in the decoder help maintain informative latent codes.
- KL-divergence weight (β in β-VAE) controls the disentanglement-reconstruction trade-off; increasing β promotes disentanglement at the cost of reconstruction quality.
- *Interview-level:* Derive the ELBO from the marginal likelihood using Jensen's inequality. What is the reparameterization trick, and why is it necessary for gradient estimation?

16.2 Generative Adversarial Networks (GAN)

Generative Adversarial Networks were introduced by Goodfellow et al. (2014) as a fundamentally new approach to generative modeling, framed as a two-player minimax game between a generator network (producing synthetic data) and a discriminator network (distinguishing real from synthetic data). The key motivation was to sidestep the intractability of maximum likelihood estimation for deep generative models: rather than explicitly modeling the data density (as VAEs approximate via the ELBO), GANs implicitly learn a data distribution by training the generator to fool the discriminator. This adversarial training signal proved remarkably effective at capturing fine-grained, perceptually realistic structure in image data—producing samples of unprecedented visual quality—making GANs the dominant generative model for images throughout 2014–2021 before diffusion models surpassed them.

✓ Pros

- Produces sharp, high-fidelity samples; state-of-the-art image generation for over a decade.
- No explicit likelihood model required; learns to generate via adversarial feedback.
- Flexible; applicable to images, text-to-image, video, and audio generation.

✗ Cons

- Training instability: mode collapse, oscillation, and non-convergence are endemic.
- No tractable likelihood; cannot be used for density estimation.
- Mode dropping: GAN may ignore modes of the true distribution.
- Hyperparameter and architecture sensitivity is extreme.
- Largely superseded by diffusion models for high-quality unconditional generation.

▷ Technical Use Case

Use when the primary objective is high-perceptual-quality sample generation and not density estimation. Conditional GANs (cGAN, pix2pix, StyleGAN) are most effective in practice. Wasserstein GAN (WGAN-GP) provides training stability via a principled distance metric and gradient penalty. Diffusion models are the current state-of-the-art alternative for most generation tasks.

★ Challenges & Advanced Considerations

- Mode collapse detection and mitigation: minibatch discrimination, unrolled GANs, and spectral normalization address mode collapse through different mechanisms.
- Evaluation metrics: FID (Fréchet Inception Distance) measures distributional similarity; IS (Inception Score) measures per-sample diversity and quality; neither is a substitute for human evaluation.
- *Interview-level:* What is the theoretical basis for training instability in the original GAN? How does the Wasserstein distance (Earth Mover's distance) address vanishing gradients in discriminator training?

16.3 Diffusion Models (DDPM, Score Matching)

Diffusion probabilistic models were introduced by Sohl-Dickstein et al. (2015) and made practically effective by Ho, Jain, and Abbeel (2020) with DDPM (Denoising Diffusion Probabilistic Models), subsequently connected to score matching by Song et al. (2020) via the score-based generative modeling framework. The motivation was to define a generative model as the reversal of a fixed forward Markov chain that gradually corrupts data with Gaussian noise over many steps until the data becomes pure noise—a process whose reversal can be learned by a neural network trained to predict the noise at each step. Unlike GANs, diffusion models have a stable, non-adversarial training objective and achieve superior mode coverage; unlike VAEs, they do not require a learned encoder and produce sharper samples. These properties made diffusion models the state-of-the-art generative framework for images, audio, and video generation as of 2022–2024.

✓ Pros

- Current state-of-the-art for image, audio, and video generation; superior sample quality and diversity vs. GANs.
- Training is stable; no adversarial game; optimizes a simple denoising objective.
- Mode coverage: diffusion models sample from the full data distribution without mode dropping.
- Conditional generation via classifier-free guidance is flexible and powerful.

✗ Cons

- Sampling is slow: requires many denoising steps (50–1000); latent diffusion models mitigate via compression.
- Computationally expensive training; requires large GPU clusters for state-of-the-art models.
- Less suited for discrete data (text) than continuous data (images); discrete diffusion models exist but are less mature.

▷ Technical Use Case

Use for high-quality image, audio, and video synthesis. Latent diffusion models (LDM, Stable Diffusion) perform the diffusion process in a compressed latent space, enabling faster sampling and lower memory requirements. DDIM sampling enables deterministic sampling in ∼50 steps without quality loss.

★ Challenges & Advanced Considerations

- Guidance scale in classifier-free guidance: higher guidance increases sample-prompt alignment but reduces diversity; trade-off must be calibrated for the application.
- The number of diffusion steps, noise schedule, and architecture (U-Net, DiT) are structural choices with significant performance implications.
- *Interview-level:* Derive the DDPM training objective as a variational lower bound. What is the score function, and how does score matching relate to denoising diffusion?

Additional Foundational Methods

17.1 Conformal Prediction

Conformal prediction was developed by Vovk, Gammerman, and Shafer (1999, 2005) within the algorithmic learning theory framework as a method for producing prediction sets with finite-sample, distribution-free validity guarantees. The central motivation was to move beyond asymptotic or distributional coverage guarantees of classical prediction intervals, which are only valid as $n \to \infty$ or under specific parametric assumptions, toward guarantees that hold for any sample size, any model, and any data distribution satisfying only exchangeability. Conformal prediction wraps any point predictor with a calibration procedure that uses a held-out calibration set to set a threshold on a non-conformity score, ensuring that the resulting prediction set contains the true label with a user-specified probability—a rigorous, practical solution to the uncertainty quantification problem in ML that has seen a surge of interest in the safety-critical and regulatory ML communities.

✓ Pros

- Distribution-free prediction intervals with guaranteed marginal coverage at any confidence level.
- Model-agnostic; wraps any point predictor to produce valid uncertainty sets.
- No distributional assumptions required; coverage holds under exchangeability.
- Split conformal is computationally trivial; full conformal requires retraining per test point.

✗ Cons

- Marginal coverage guarantee only; conditional coverage (for subgroups) is not guaranteed.
- Calibration set must be held out; reduces training data available.
- Assumes exchangeability; fails under distribution shift (covariate shift).
- Prediction intervals can be overly conservative if the non-conformity score is poorly designed.

▷ Technical Use Case

Use whenever calibrated, finite-sample uncertainty quantification is required without distributional assumptions. Preferred over asymptotic confidence intervals in small samples or

under distribution uncertainty. Mondrian conformal prediction (stratified by covariate group) achieves conditional coverage for specific subgroups.

★ Challenges & Advanced Considerations

- Non-conformity score design is critical; a poor score yields valid but uninformative (very wide) intervals.
- Weighted conformal prediction extends coverage guarantees to covariate shift settings via importance weighting.
- *Interview-level:* Prove the marginal coverage guarantee of split conformal prediction. What is the connection between conformal prediction and the quantiles of the non-conformity score distribution?

17.2 Federated Learning

Federated Learning was introduced by McMahan et al. at Google (2017) as a distributed machine learning paradigm designed to train models across many decentralized devices or institutions without centralizing raw data, motivated by growing privacy regulations (GDPR, HIPAA) and the practical impossibility of pooling sensitive user data (e.g., mobile keyboard data, medical records). The key innovation of FedAvg was to have each participating client locally compute multiple gradient steps on its private data and send only model weight updates (not raw data) to a central server, which aggregates the updates by a weighted average to produce a new global model. This privacy-by-design approach enables collaborative learning across data silos that cannot be merged—at the cost of new challenges around non-IID data distribution, communication efficiency, and the residual privacy risks of gradient inversion attacks.

✓ Pros

- Trains models across distributed data sources without centralizing raw data; privacy-preserving by design.
- Enables learning from sensitive or proprietary datasets that cannot be pooled.
- FedAvg: simple and effective aggregation via weighted averaging of client model updates.

✗ Cons

- Non-IID data across clients degrades convergence; local updates drift from the global optimum (client drift).
- Communication rounds are the bottleneck; each round requires a full forward/backward pass on clients.
- Privacy guarantees of basic FL are limited; gradient inversion attacks can reconstruct training data.
- Heterogeneous client compute and availability (system heterogeneity) complicates synchronous training.

▷ Technical Use Case

Use when data cannot be centralized due to regulatory, privacy, or proprietary constraints. Differential privacy (DP-SGD) provides formal privacy guarantees at the cost of model accuracy. Secure aggregation prevents the server from inspecting individual client updates.

★ Challenges & Advanced Considerations

- Data heterogeneity (non-IID): FedProx, SCAFFOLD, and FedNova are designed to reduce client drift compared to FedAvg.
- Privacy-utility trade-off: differential privacy noise introduced for privacy guarantees degrades model accuracy; the privacy budget ϵ must be chosen carefully.
- *Interview-level:* What is gradient inversion, and why does it imply that basic federated learning does not guarantee privacy? How does differential privacy address this?

17.3 Transfer Learning & Domain Adaptation

Transfer learning addresses a fundamental limitation of standard supervised learning: the requirement for large, labeled task-specific datasets that are expensive or impossible to acquire for every new problem. The core insight, rooted in the inductive transfer concept from cognitive science, is that knowledge learned while solving one task (source) can be reused to accelerate and improve learning on a related task (target) with limited labeled data. In deep learning, this is operationalized by pretraining a model on a large source dataset and then fine-tuning its parameters on the target task, with the pretrained representations providing a rich starting point. Domain adaptation specifically addresses the distributional shift between source and target domains, proposing methods (adversarial alignment, importance weighting) that force the learned representations to be invariant to domain-specific variation while retaining task-relevant structure.

✓ Pros

- Leverages pretrained representations; dramatically reduces labeled data requirements for downstream tasks.
- Foundation models (LLMs, Vision Transformers) provide powerful priors across diverse tasks.
- Domain adaptation bridges distribution gap between source and target domains.

✗ Cons

- Negative transfer: when source and target domains are sufficiently dissimilar, pretraining can hurt.
- Catastrophic forgetting: fine-tuning on a target task erases source task knowledge.
- Domain gap estimation is non-trivial; degree of distribution shift is often unknown.
- Fine-tuning large foundation models requires significant compute and engineering infrastructure.

▷ Technical Use Case

Use when target domain labeled data is scarce but a pretrained model on a related (or large-scale generic) domain is available. Feature extraction (frozen encoder) vs. full fine-tuning vs. parameter-efficient fine-tuning (LoRA, adapters) must be selected based on domain similarity and compute budget. Domain-adversarial training (DANN) forces domain-invariant representations for unsupervised domain adaptation.

★ Challenges & Advanced Considerations

- Layer-wise learning rate decay: lower layers of a pretrained network should be updated more slowly than higher layers; flat fine-tuning learning rates risk disrupting low-level

features.
- Distribution shift types: covariate shift, label shift, and concept shift require different adaptation strategies.
- *Interview-level:* What is the theoretical connection between domain adaptation and the H-divergence? What does the domain adaptation bound (Ben-David et al.) imply about the feasibility of transfer across distant domains?

17.4 Neural Architecture Search (NAS)

Neural Architecture Search was introduced by Zoph and Le (2017) as a method for automating the design of neural network architectures, motivated by the observation that architecture design is a critical but labor-intensive and expert-dependent component of deep learning pipelines. Hand-designing architectures requires extensive empirical experimentation and domain expertise; NAS proposes to replace this human effort with an automated search over a defined architecture space using a search algorithm (reinforcement learning, evolutionary algorithms, or gradient-based optimization). Differentiable NAS (DARTS, Liu et al., 2019) made the search tractable by relaxing the discrete architecture selection into a continuous optimization over operation weights, enabling gradient-based joint optimization of architecture and weights on a validation set—reducing search cost from thousands of GPU-days (RL-based NAS) to a few GPU-days.

✓ Pros

- Automates architecture design; discovers architectures competitive with or superior to human-designed ones.
- Differentiable NAS (DARTS) enables gradient-based architecture search; orders of magnitude faster than RL-based search.
- Applicable to diverse architecture spaces: cell-based, layer-based, and operation-based search.

✗ Cons

- Computationally expensive; RL-based NAS requires thousands of GPU-days.
- Performance of discovered architectures is highly sensitive to the search space definition.
- DARTS is known to overfit to the validation set during architecture search; produces architectures that skip connections rather than learn.
- Generalization from proxy tasks (small datasets, few epochs) to full evaluation is unreliable.

▷ Technical Use Case

Use when resource-optimal architecture design is required and training compute can be invested in architecture search. One-shot NAS with weight sharing (single-path NAS) is the practical approach; differentiable NAS is the computationally tractable research variant. Most practitioners benefit from established architectures rather than running NAS from scratch.

★ Challenges & Advanced Considerations

- DARTS collapse: the learned architecture selects skip connections because they have zero gradient magnitude at initialization; regularization (DARTS+, SDARTS) addresses this.
- The choice of search space is the dominant factor; a well-designed search space is more important than the search algorithm.

- *Interview-level:* What is the weight-sharing assumption in one-shot NAS, and why can it lead to incorrect architecture rankings?

17.5 Survival Analysis (Cox PH, Kaplan-Meier)

Survival analysis is a branch of statistics developed for the analysis of time-to-event data, with Kaplan and Meier's (1958) non-parametric estimator of the survival function and Cox's (1972) proportional hazards regression model being the two most foundational contributions. These methods were developed to handle a unique feature of time-to-event data: censoring—the event of interest (death, failure, churn) is not observed for all subjects within the study period, yet simply discarding censored observations would introduce severe selection bias. The Kaplan-Meier estimator incorporates censoring into a non-parametric survival curve estimate; the Cox model extends this to regression by modeling the hazard function as a product of a baseline hazard (unspecified) and an exponential function of covariates, enabling covariate-adjusted hazard ratio estimation without assuming a parametric form for the time distribution.

✓ Pros

- Explicitly models time-to-event data with censoring; handles incomplete observations correctly.
- Cox PH: semi-parametric, estimates hazard ratios without specifying the baseline hazard.
- Kaplan-Meier: non-parametric survival function estimation; no distributional assumption.
- Well-established statistical theory; confidence intervals and log-rank tests are standard.

✗ Cons

- Cox PH assumes proportional hazards (PH); hazard ratio is constant over time. Violation leads to biased estimates.
- Does not predict individual survival times; estimates hazard ratios and population-level survival curves.
- Deep survival models (DeepSurv, DRSA) relax PH assumption but lose interpretability.
- Non-informative censoring assumption: censoring must be independent of survival time.

▷ Technical Use Case

Use when outcomes are time-to-event with censoring. Cox PH is appropriate when the proportional hazards assumption holds (verifiable via scaled Schoenfeld residuals). Accelerated failure time (AFT) models are preferred when the PH assumption is violated. Deep survival models apply when covariate effects are complex and non-linear.

★ Challenges & Advanced Considerations

- PH assumption testing via log-log plots and Schoenfeld residual tests must be performed; time-varying coefficients via Cox time-interaction terms address PH violations.
- Informative censoring (dropout correlated with outcome) is a critical untestable assumption; sensitivity analysis is required.
- *Interview-level:* Derive the partial likelihood for the Cox model. Why does the partial likelihood eliminate the baseline hazard, and what is the consequence for the estimator's efficiency compared to full likelihood methods?

Document Summary: Decision Framework

Model Selection Heuristics

Tabular Supervised Learning: Start with gradient boosting (XGBoost/LightGBM). Add regularized linear model for interpretability baseline. Use neural networks only when $n > 10^5$ or transfer learning applies.

High-Dimensional Sparse Settings: Lasso or L_1-regularized logistic regression. If groups of correlated features, use Elastic Net. If $n \ll p$ and interactions matter, use Random Forest with feature subsampling.

Sequence Data: ARIMA for short, stationary, univariate series with required uncertainty intervals. LSTM/TCN for complex, multivariate, long series. Transformers for long-range dependency with sufficient data.

Unstructured Data: Always start from a pretrained foundation model. Fine-tune with PEFT (LoRA, adapters) before full fine-tuning. Domain-specific pretraining beats generic when domain shift is large.

Causal Questions: Define estimand first (ATE, ATT, LATE). Select identification strategy based on available design (RCT > DiD > RDD > IV > Propensity Score). Never interpret associational models causally.

Uncertainty Quantification: Conformal prediction for distribution-free coverage. GP for small-n function estimation. BNN/Deep Ensembles for neural network uncertainty. Always separate epistemic and aleatoric uncertainty.

Explainability: TreeSHAP for tree-based models. Kernel SHAP or LIME for general models. PDP/ALE for global marginal effects. Always explain the *model*, not the *world* — communication is critical.

Universal Anti-Patterns

- Fitting PCA / t-SNE / UMAP on the full dataset before splitting train/test (data leakage).
- Using model-derived feature importance (MDI) for causal attribution.
- Interpreting SHAP values as causal effects.
- Selecting the number of clusters post-hoc to confirm a hypothesis.
- Applying ARIMA to a non-stationary series without differencing or testing for unit roots.
- Using accuracy as the primary metric for imbalanced classification.
- Reporting only training loss / training AUC without held-out evaluation.
- Conflating statistical significance with practical significance.
- Ignoring distributional shift between training and deployment environments.
- Comparing models on metrics inconsistent with the decision-relevant loss function.

This document is a living technical reference. Methods, best practices, and state-of-the-art recommendations evolve with the field.