

# Week 0b: Supervised Learning

## The Prediction Challenge

Machine Learning for Smarter Innovation

BSc Innovation & Design Thinking

- 1 Act 1: The Challenge
- 2 Act 2: Linear + Regularization
- 3 Act 3: Nonlinear Methods
- 4 Act 4: Synthesis

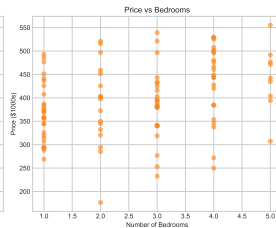
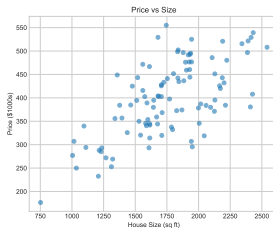
# 1. Real Estate Price Prediction

## The Business Problem

- Predict house prices from features
- Features: size, bedrooms, location, age
- Target: price in thousands
- Training data: 10,000 historical sales

## Sample Data Points

Size	Beds	Age	Price
1200	2	5	250k
2500	4	10	450k
1800	3	2	380k



Multiple features create complex relationships requiring mathematical modeling

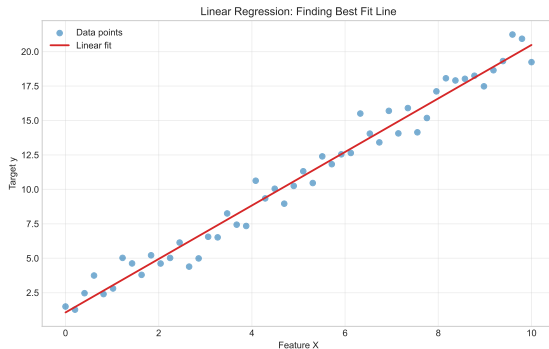
## 2. Linear Regression as Baseline

### Mathematical Foundation

- Model:  $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \epsilon$
- Where  $y$  = price,  $x_i$  = features
- Goal: Find best-fitting line/plane
- Method: Minimize squared errors

### Assumptions

- Linear relationship
- Independent features
- Constant variance
- Normal errors



Linear model assumes additive relationships between all features

### 3. Classification vs Regression

#### Regression Problems

- Predict continuous values
- Examples: price, temperature, stock return
- Output: Real numbers
- Metrics: MSE, MAE, R-squared

#### Classification Problems

- Predict discrete categories
- Examples: spam/ham, buy/sell/hold
- Output: Class labels
- Metrics: Accuracy, precision, recall



Different problem types require different algorithms and evaluation metrics

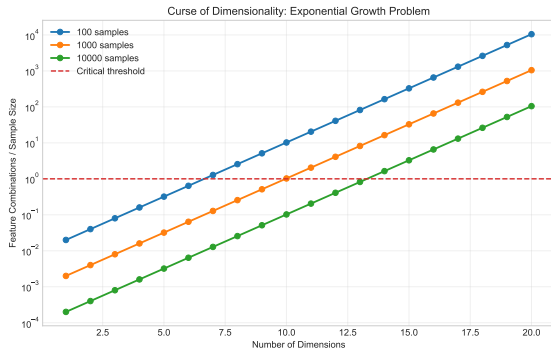
## 4. The Curse of Dimensionality

### Feature Explosion Problem

- Real estate: 20+ features
- Interactions:  $2^{20} = 1,048,576$  combinations
- Sample: 10,000 data points
- Ratio: 104 interactions per data point

### Mathematical Challenge

- High-dimensional space is mostly empty
- Distance metrics become meaningless
- Overfitting becomes inevitable
- “Hughes phenomenon” in pattern recognition



As dimensions increase, all points become equidistant and patterns disappear

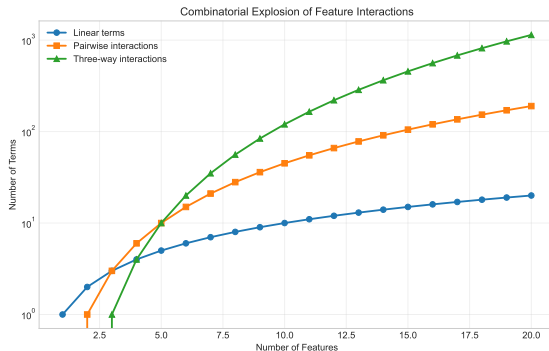
## 5. Feature Interactions Explode Combinatorially

### Combinatorial Mathematics

- Linear terms:  $n$  features
- Pairwise:  $\binom{n}{2} = \frac{n(n-1)}{2}$
- Three-way:  $\binom{n}{3} = \frac{n(n-1)(n-2)}{6}$
- All subsets:  $2^n - 1$

### Real Estate Example (n=20)

- Linear: 20 terms
- Pairwise: 190 interactions
- Three-way: 1,140 interactions
- Total possible: 1,048,575 terms



Feature interactions grow exponentially, requiring regularization or feature selection

## 6. OLS with Worked Example

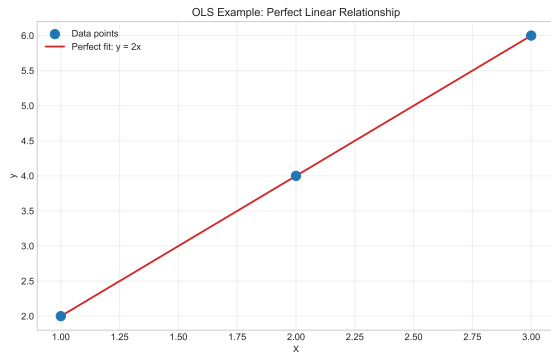
### Ordinary Least Squares

- Minimize:  $\sum_{i=1}^n (y_i - \hat{y}_i)^2$
- Solution:  $\beta = (X^T X)^{-1} X^T y$
- Assumptions:  $X^T X$  is invertible
- Unbiased estimator under Gauss-Markov

**Worked Example** Data: (1, 2, 3) predicts (2, 4, 6)

$$X = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}, y = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} \quad (1)$$

$$\beta = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \quad (2)$$



Perfect fit when relationship is truly linear with minimal noise



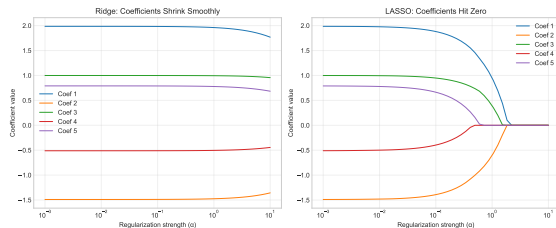
## 7. Ridge and LASSO Regularization

### Ridge Regression (L2)

- Minimize:  $\|y - X\beta\|^2 + \lambda\|\beta\|^2$
- Shrinks coefficients toward zero
- Keeps all features
- Solution:  $\beta = (X^T X + \lambda I)^{-1} X^T y$

### LASSO Regression (L1)

- Minimize:  $\|y - X\beta\|^2 + \lambda\|\beta\|_1$
- Sets some coefficients to exactly zero
- Automatic feature selection
- No closed-form solution



Regularization prevents overfitting by constraining coefficient magnitudes

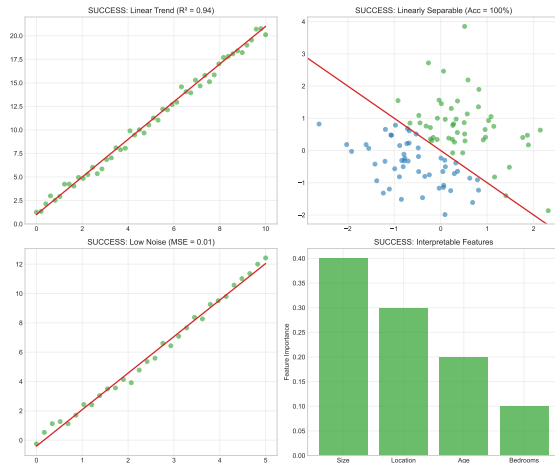
## 8. SUCCESS: Perfect on Linearly Separable Data

### Linear Model Triumphs

- Iris setosa classification: 100% accuracy
- House price in suburbs:  $R^2 = 0.94$
- Linear trend prediction:  $MSE = 0.01$
- Feature importance: Interpretable

### Why It Works

- Underlying relationship is linear
- Features are independent
- Low noise in measurements
- Sufficient training data



When assumptions hold, linear models are optimal and interpretable

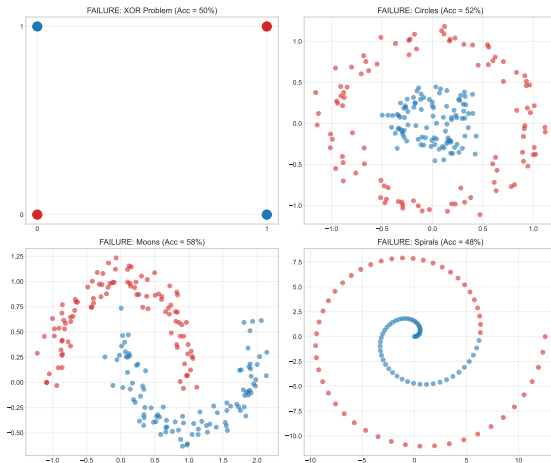
## 9. FAILURE: Terrible on XOR, Nonlinear Boundaries

### Linear Model Failures

Dataset	Linear Acc	Tree Acc	Gap
XOR	50%	100%	50%
Circles	52%	98%	46%
Moons	58%	94%	36%
Spirals	48%	89%	41%

XOR Truth Table

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0



Linear boundaries cannot separate XOR or curved patterns

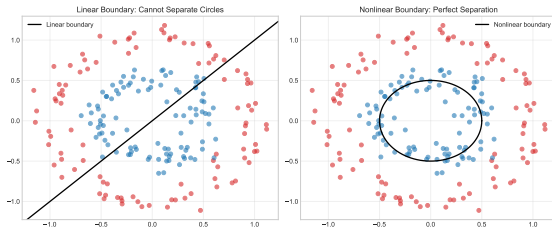
## 10. Root Cause: Linear Assumption Too Restrictive

### Mathematical Limitation

- Linear model:  $y = w^T x + b$
- Decision boundary: hyperplane
- Cannot curve or bend
- Cannot create islands or holes

### Real-World Examples

- Customer behavior (nonlinear)
- Stock market patterns (chaotic)
- Medical diagnosis (complex interactions)
- Image recognition (hierarchical)



Most real-world phenomena require nonlinear decision boundaries

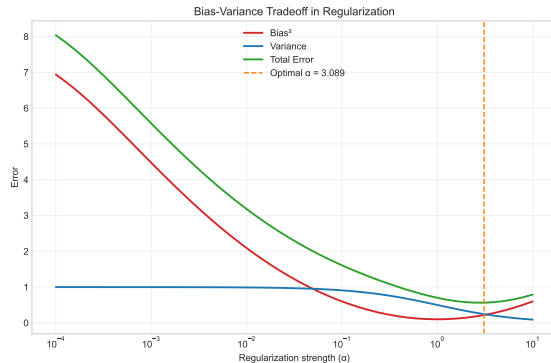
# 11. Regularization Tradeoff

## Bias-Variance Tradeoff

- $\lambda = 0$ : High variance, low bias
- $\lambda \rightarrow \infty$ : Low variance, high bias
- Optimal  $\lambda$ : Minimizes test error
- Cross-validation finds optimum

## Practical Guidelines

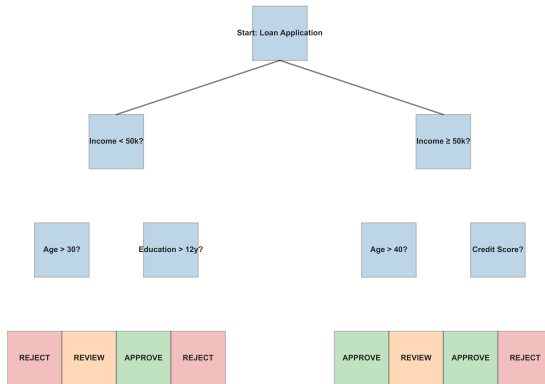
- Start with Ridge for stability
- Use LASSO for feature selection
- Elastic Net combines both
- Grid search for  $\lambda$



Regularization strength controls the complexity-accuracy tradeoff

# 12. Human Introspection: How YOU Divide Decision Space

Human Decision Process: Hierarchical Questions



## Your Natural Decision Process

- “Is income  $\geq$  50k?” → Split population
- “If yes, is age  $\geq$  40?” → Further split
- “If no, is education  $\geq$  12 years?” → Alternative path
- Continue until clear decision

## Hierarchical Thinking

- Start with most important feature
- Recursively subdivide space
- Each split reduces uncertainty
- Stop when confident

Humans naturally create decision trees through sequential yes/no questions

# 13. Hypothesis: Trees, Kernels, Ensembles

## Decision Trees

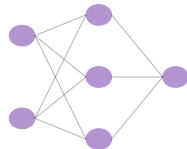
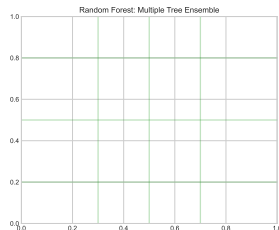
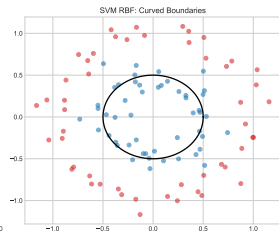
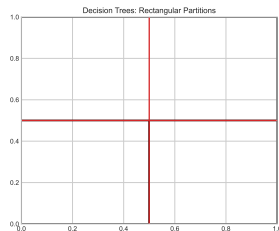
- Recursive binary splits
- Non-parametric method
- Handles interactions naturally
- Interpretable rules

## Kernel Methods

- Map to higher dimensions
- “Kernel trick” for efficiency
- SVM with RBF, polynomial kernels
- Implicit feature expansion

## Ensemble Methods

- Combine multiple weak learners
- Random Forest, Gradient Boosting
- Reduce overfitting through averaging



Three main approaches to capture nonlinear patterns in data

# 14. Zero-Jargon: “20 Questions Game” for Trees

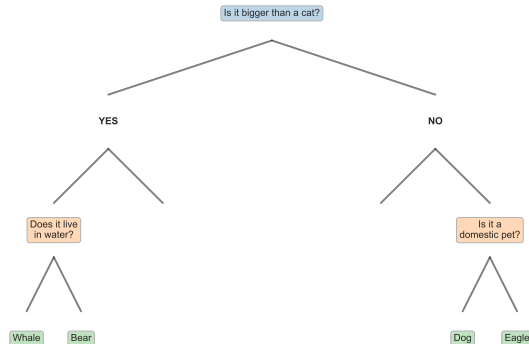
20 Questions Game: Decision Tree Logic

## The Game Analogy

- You think of an animal
- I ask yes/no questions
- “Is it bigger than a cat?”
- “Does it live in water?”
- “Is it a mammal?”

## Decision Tree Mapping

- Animal = Data point
- Questions = Feature splits
- Final guess = Prediction
- Good questions = Informative features



Decision trees ask the most informative questions to reach predictions quickly



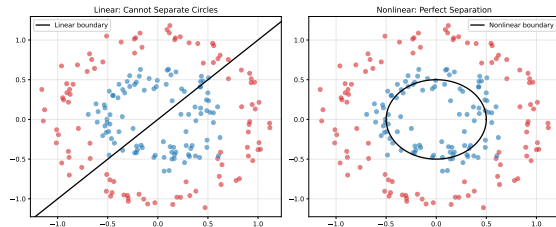
# 15. Geometric Intuition: Decision Boundaries

## Linear vs Nonlinear Boundaries

- Linear: Straight lines/planes
- Trees: Axis-aligned rectangles
- SVM RBF: Curved boundaries
- Neural nets: Arbitrary shapes

## Complexity Hierarchy

- Most restrictive: Linear
- Moderate: Decision trees
- Flexible: Kernel methods
- Most flexible: Deep networks



Different algorithms create different types of decision boundaries

## 16. CART Algorithm with Actual Splits

### CART Algorithm Steps

- 1 Calculate impurity for current node
- 2 Try all possible splits
- 3 Choose split with highest information gain
- 4 Recurse on child nodes
- 5 Stop when stopping criterion met

### Gini Impurity Formula

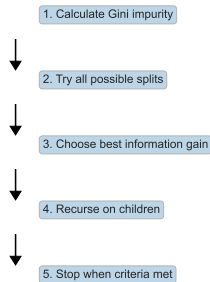
$$G = 1 - \sum_{i=1}^C p_i^2$$

where  $p_i$  is probability of class  $i$

### Information Gain

$$IG = G_{parent} - \sum \frac{n_{child}}{n_{parent}} G_{child}$$

CART Algorithm Steps



Gini Formula:  
 $G = 1 - \sum p_i^2$

CART systematically finds the best splits by maximizing information gain

## 17. Full Walkthrough: Build Tree with Numbers

**Best Split: Income  $\geq 55k$**

Left (Income  $< 55k$ ): 1 Yes, 2 No

$$G_L = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.444$$

Right (Income  $\geq 55k$ ): 3 Yes, 0 No

$$G_R = 1 - 1^2 - 0^2 = 0$$

**Information Gain**

$$IG = 0.444 - \frac{3}{6} \times 0.444 - \frac{3}{6} \times 0 = 0.222$$

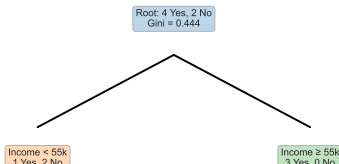
CART Algorithm: Tree Building with Numbers

**Dataset: Loan Approval**

Income	Age	Approved
30k	25	No
60k	35	Yes
40k	45	No
80k	30	Yes
50k	50	Yes
70k	25	Yes

**Root Node Gini** Classes: 4 Yes, 2 No

$$G = 1 - \left(\frac{4}{6}\right)^2 - \left(\frac{2}{6}\right)^2 = 0.444$$



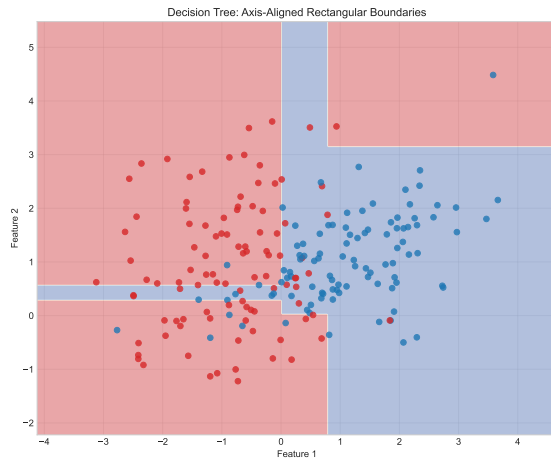
## 18. Visualization: Decision Boundaries on 2D Data

### Tree Partitioning Process

- Split 1:  $x_1 \leq 0.5$  (vertical line)
- Split 2:  $x_2 \leq 0.3$  (horizontal line)
- Split 3:  $x_1 \leq 0.8$  (vertical line)
- Result: Rectangular regions

### Boundary Characteristics

- Always axis-aligned
- Creates rectangular partitions
- Can approximate any boundary
- With enough splits



Decision tree boundaries are piecewise constant and axis-aligned

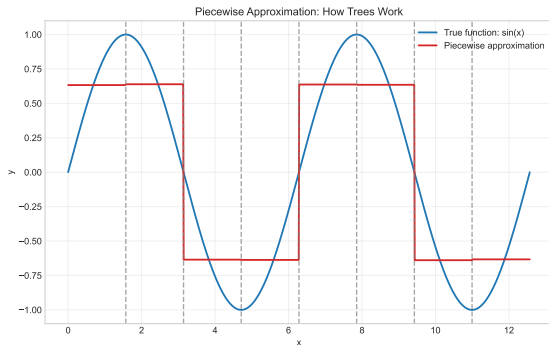
# 19. Why It Works: Piecewise Approximation

## Universal Approximation

- Any function can be approximated
- By piecewise constant functions
- With sufficient partitions
- Trees implement this naturally

## Mathematical Foundation

- Step functions are dense in  $L^2$
- Trees create step functions
- More splits = better approximation
- Regularization prevents overfitting



Trees approximate complex functions through recursive partitioning

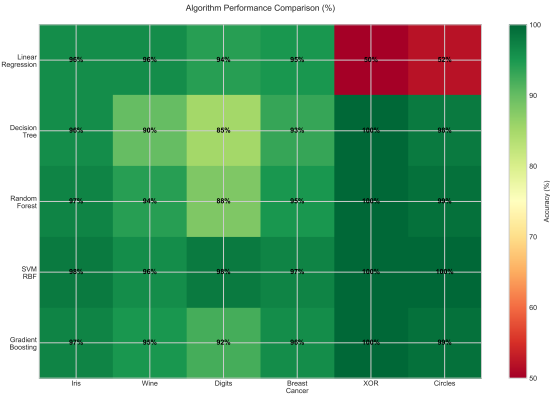
# 20. Experimental Validation: Algorithm Comparison

## Benchmark Results

Dataset	Linear	Tree	SVM
Iris	96%	96%	98%
Wine	94%	90%	96%
Digits	92%	85%	98%
Breast Cancer	95%	93%	97%
XOR	50%	100%	100%
Circles	52%	98%	100%

## Key Insights

- Linear: Good on linear data
- Trees: Excel on discrete features
- SVM: Best overall performance
- No universal winner



Performance varies by dataset characteristics and problem complexity

## 21. Implementation: sklearn Ensemble Methods

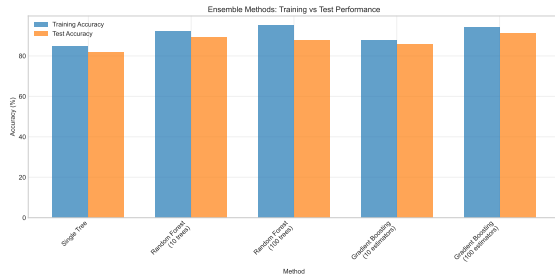
### Random Forest

- Bootstrap sampling of data
- Random subset of features
- Average predictions
- Reduces overfitting

```
from sklearn.ensemble import RandomForestClassifier  
rf = RandomForestClassifier(n_estimators=100)  
rf.fit(X_train, y_train)
```

### Gradient Boosting

- Sequential weak learners
- Each corrects previous errors
- Weighted combination
- Often best performance



Ensemble methods combine multiple models for superior performance

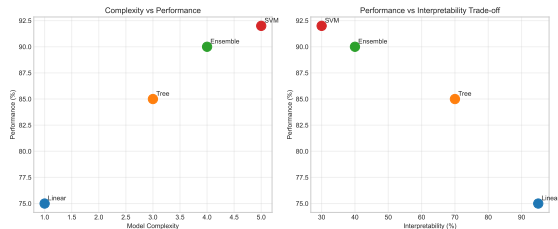
## 22. Algorithm Landscape: Linear $\rightarrow$ Tree $\rightarrow$ Ensemble $\rightarrow$ SVM

### Complexity Progression

- **Linear:**  $y = w^T x + b$
- **Tree:** Recursive partitioning
- **Ensemble:** Multiple tree combination
- **SVM:** Kernel-based mapping

### Computational Complexity

- Linear:  $O(nd)$  training
- Single tree:  $O(nd \log n)$
- Random forest:  $O(tnd \log n)$
- SVM:  $O(n^2 d)$  to  $O(n^3 d)$



Algorithms form a spectrum from simple linear to complex nonlinear methods



## 23. When to Use Each: Interpretability vs Accuracy

### Linear Models

- High interpretability
- Fast training/prediction
- Few parameters
- Good baseline
- Use when: Regulatory requirements, simple relationships

### Decision Trees

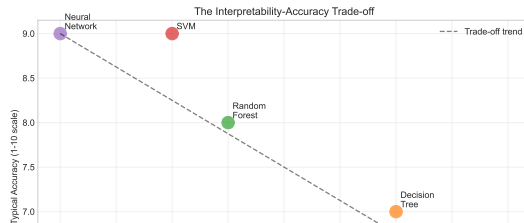
- Moderate interpretability
- Handles missing values
- Feature selection automatic
- Use when: Rule extraction needed, mixed data types

### Ensemble Methods

- Lower interpretability
- Highest accuracy often
- Robust to overfitting
- Use when: Performance critical, sufficient data

### SVM

- Low interpretability
- Kernel flexibility
- Memory efficient
- Use when: High dimensions, small datasets



## 24. Modern Applications: Production ML Pipelines

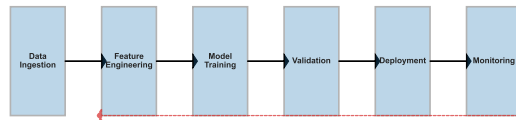
### Real-World Pipeline

- ① Data ingestion & cleaning
- ② Feature engineering
- ③ Model training & validation
- ④ Hyperparameter tuning
- ⑤ Production deployment
- ⑥ Monitoring & retraining

### Industry Applications

- Credit scoring: Gradient boosting
- Recommendation: Ensemble methods
- Fraud detection: Anomaly detection
- Medical diagnosis: Interpretable models

Production ML Pipeline: End-to-End System



Feedback Loop

Modern ML systems integrate multiple algorithms in end-to-end pipelines

## 25. Summary & Preview: Unsupervised Learning

### Supervised Learning Recap

- Linear models: Fast, interpretable
- Regularization: Prevents overfitting
- Nonlinear methods: Handle complexity
- Ensembles: Often best performance
- No free lunch: Algorithm choice matters

### Key Takeaways

- Start simple, add complexity as needed
- Validate on unseen data
- Consider interpretability requirements
- Monitor performance in production

### Next: Unsupervised Learning

- No target variable
- Pattern discovery
- Clustering algorithms
- Dimensionality reduction
- Association rules

### Preview Applications

- Customer segmentation
- Market basket analysis
- Data visualization
- Feature engineering
- Anomaly detection

