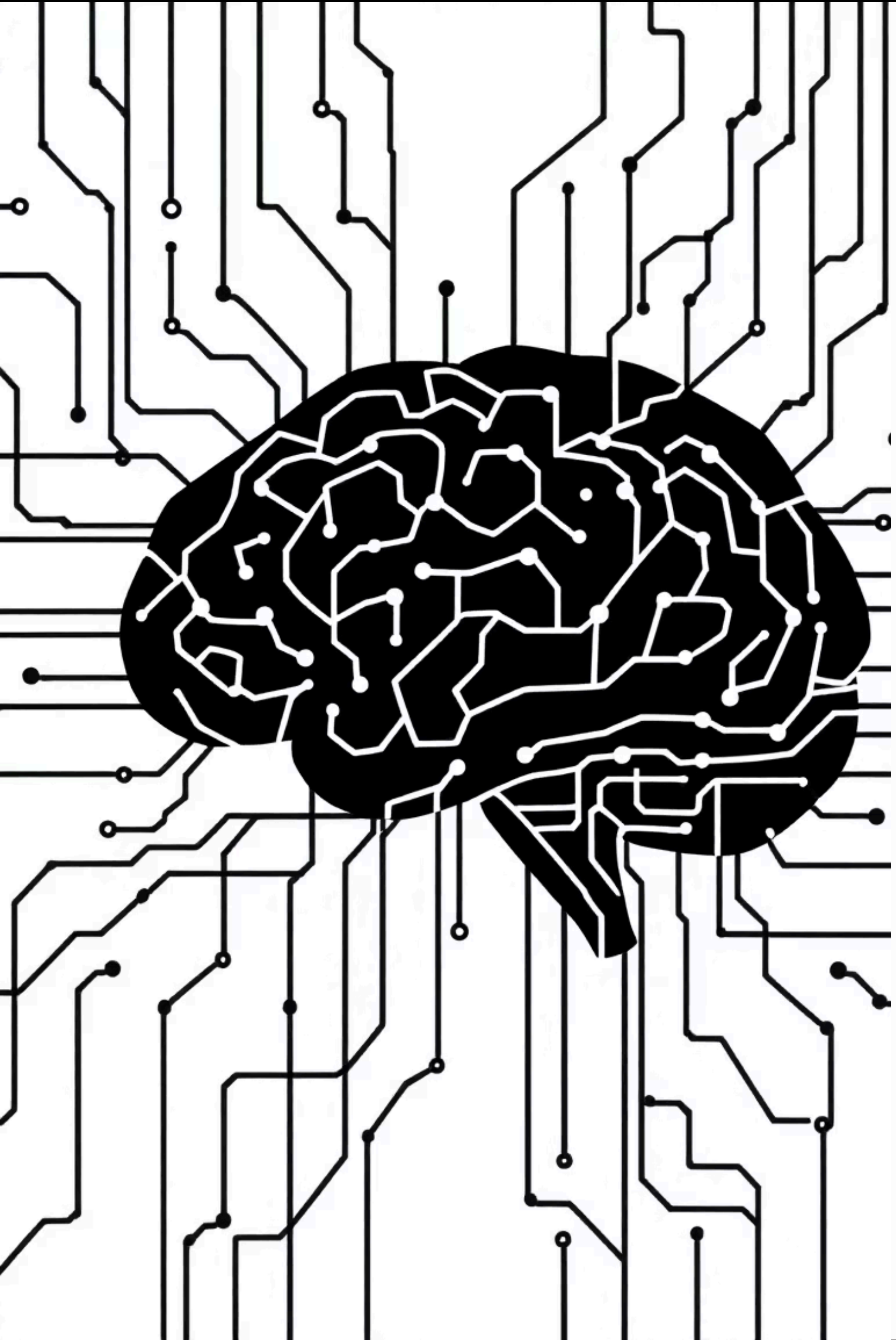


Classification in Machine Learning

Understanding How Machines Categorize Data

Classification is a fundamental supervised learning technique where machines learn to assign categorical labels to data points. Unlike regression, which predicts continuous values, classification predicts discrete categories—transforming raw data into actionable insights.

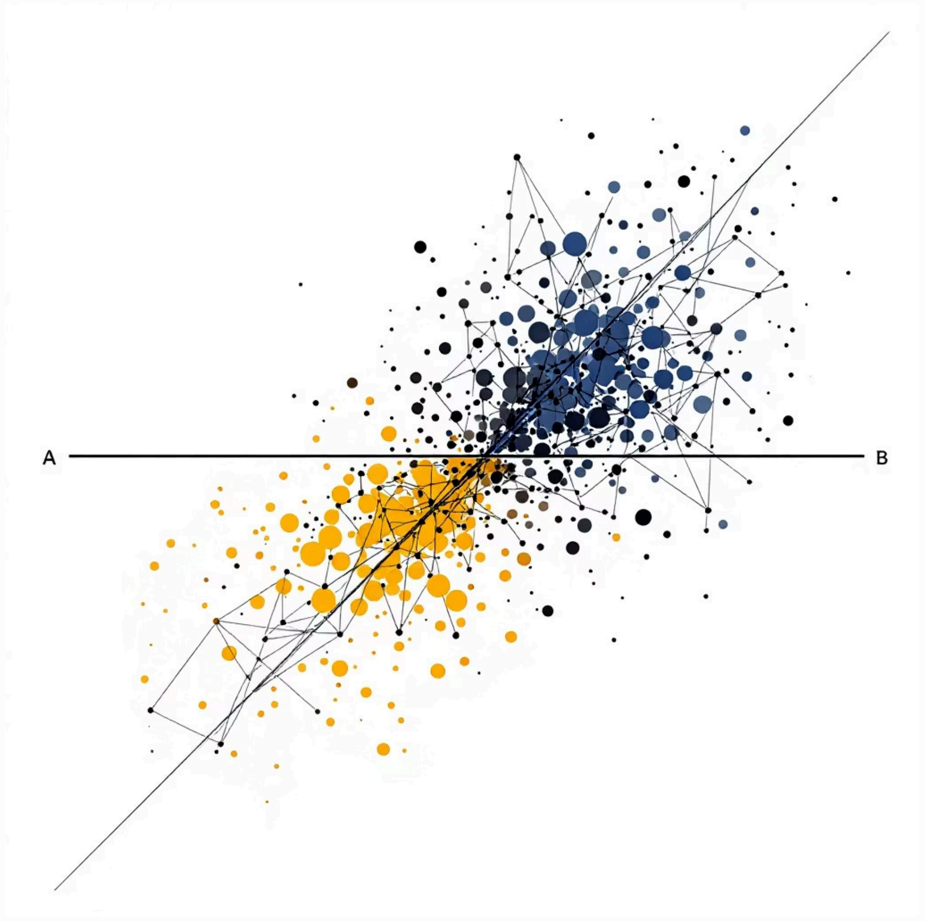
Classification = Categorizing Data

What Makes Classification Unique?

Classification algorithms predict **discrete labels** rather than continuous values. The model learns patterns from labeled training data, then applies those patterns to categorize new, unseen examples.

Real-World Applications

- **Email filtering:** Spam vs. Not Spam
- **Medical diagnosis:** Benign vs. Malignant tumor
- **Image recognition:** Cat vs. Dog vs. Bird
- **Fraud detection:** Legitimate vs. Fraudulent transaction



Key Distinction: Regression produces continuous outputs (e.g., house prices), while Classification produces discrete outputs (e.g., spam/not spam). Binary classification involves two classes; multi-class involves three or more.

The Classifier's Job: Drawing the Line

Pattern Discovery

The model analyzes training data to identify features that distinguish between classes, learning which characteristics are most predictive.

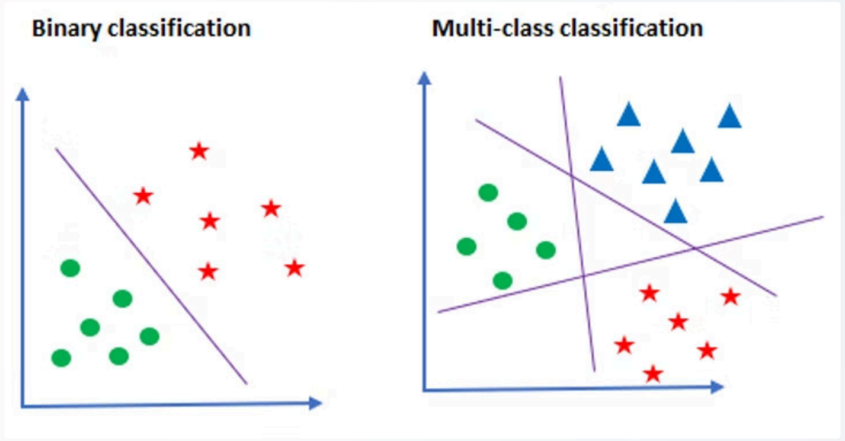
Boundary Formation

It establishes a decision boundary—a mathematical rule that separates different categories in feature space. This can be linear or non-linear.

Optimization

The algorithm iteratively refines the boundary to minimize classification errors, maximizing the model's ability to generalize to new data.

The quality of this decision boundary directly determines classification accuracy. Simple problems may require only a straight line, while complex real-world scenarios often demand sophisticated, non-linear boundaries that capture intricate patterns in high-dimensional feature spaces.



Logistic Regression: Predicting Probabilities

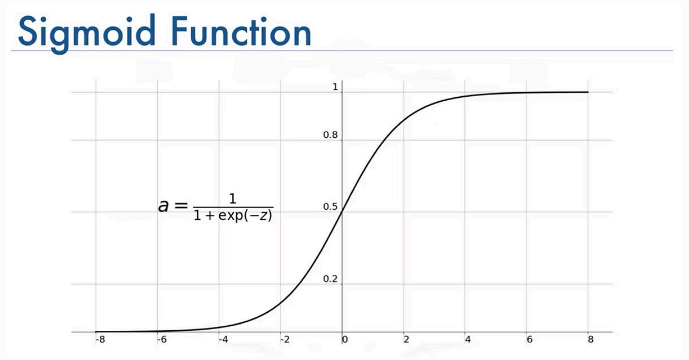
The Foundation of Binary Classification

Despite its name, Logistic Regression is a **classification algorithm**. It models the probability that an input belongs to a particular class using the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

where $z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$

The sigmoid function transforms any real-valued input into a probability between 0 and 1, creating an S-shaped curve perfect for binary decisions.



Decision Rule

If $P(y = 1|x) > 0.5$, predict **Class 1**

Otherwise, predict **Class 0**

This threshold creates a clear decision boundary in feature space, separating the two classes.

How the Model Learns to Classify

Initialize Parameters

Start with weights $w = [w_1, w_2, \ldots, w_n]$ and bias b

Make Predictions

Calculate $\hat{y} = \sigma(w^T x + b)$ for each training example

Calculate Loss

Log Loss measures error: $L = -\frac{1}{m} \sum [y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$

Update via Gradient Descent

Adjust weights: $w := w - \alpha \frac{\partial L}{\partial w}$ to minimize loss

This iterative process continues until the model converges, finding the optimal decision boundary that best separates the classes. Each iteration moves the boundary slightly to reduce classification errors, demonstrating how machines truly *learn* from their mistakes.

Handling Multiple Classes: OVA & OVO

When facing more than two categories, we extend binary classification using strategic decomposition approaches. These methods transform multi-class problems into multiple binary classification tasks.

One-vs-All (OVA / OvR)

Train K separate binary classifiers, where K = number of classes. Each classifier distinguishes one class from all others combined.

Process:

1. For each class i, create a dataset where class i is positive, all others are negative
2. Train K binary classifiers
3. For new input, run all K classifiers
4. Select class with highest confidence/probability

Complexity: K models, efficient for many classes

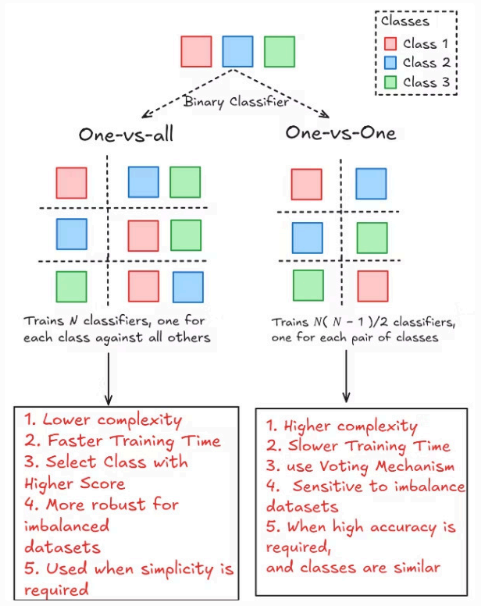
One-vs-One (OVO)

Train $\frac{K(K-1)}{2}$ binary classifiers, one for each pair of classes. Each classifier learns to distinguish between just two specific classes.

Process:

1. For each pair of classes (i, j), train a binary classifier
2. Create $\binom{K}{2}$ pairwise classifiers
3. For new input, run all pairwise classifiers
4. Use majority voting to determine final class

Complexity: More models, but each trains on smaller datasets

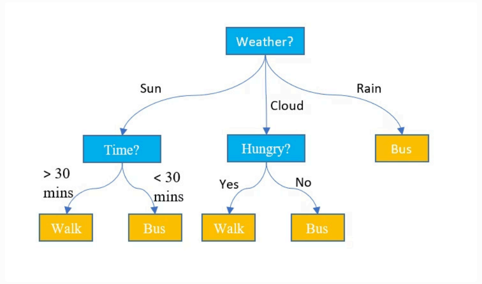

Classes
Class 1
Class 2
Class 3
Binary Classifier
One-vs-all
One-vs-One
Trains N classifiers, one for each class against all others
Trains $N(N - 1)/2$ classifiers, one for each pair of classes
1. Lower complexity
2. Faster Training Time
3. Select Class with Higher Score
4. More robust for imbalanced datasets
5. Used when simplicity is required
1. Higher complexity
2. Slower Training Time
3. use Voting Mechanism
4. Sensitive to imbalance datasets
5. When high accuracy is required, and classes are similar

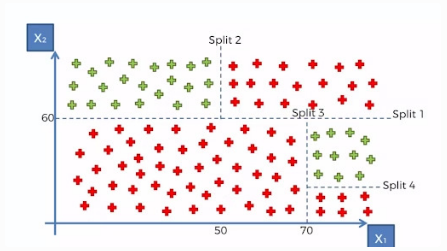
Example: For 3 classes (Cat, Dog, Rabbit), OVA trains 3 models while OVO trains 3 pairwise models. For 10 classes, OVA needs 10 models but OVO requires 45 models. Choose based on dataset size and computational resources.

Decision Trees: Learning Through Questions

Intuitive Classification

Decision Trees classify data by learning a hierarchy of yes/no questions about features. Each internal node represents a test on an attribute, each branch represents an outcome, and each leaf node represents a class label.





Key Concepts:

- **Splitting criterion:** Choose features that best separate classes
- **Gini Impurity:** $G = 1 - \sum_{i=1}^{C} p_i^2$
- **Entropy:** $H = -\sum_{i=1}^{C} p_i \log_2(p_i)$
- **Information Gain:** Reduction in entropy after split

Mathematical Foundation: Choosing the Best Split

At each node, the algorithm evaluates all possible feature splits and selects the one that maximizes information gain or minimizes Gini impurity:

$$IG(D, A) = H(D) - \sum_{v \in values(A)} \frac{|D_v|}{|D|} H(D_v)$$

where D is the dataset, A is the attribute, and D_v are the subsets after splitting on attribute A.

✓ Advantages

- Highly interpretable—visualize decision process
- Handles non-linear relationships naturally
- Requires minimal data preprocessing
- Works with numerical and categorical data

⚠️ Limitations

- Prone to overfitting with deep trees
- Sensitive to small data variations
- May create biased trees with imbalanced data
- Greedy algorithm—not guaranteed optimal

Evaluating Classification Models

Accurate performance measurement is critical for understanding model quality and identifying areas for improvement. Multiple metrics provide different perspectives on classifier behavior.

The Confusion Matrix Foundation

The confusion matrix is a 2×2 table for binary classification (or $K \times K$ for multi-class) comparing predicted vs. actual labels:

	Predicted Positive	**Predicted Negative**	**Total**
Actual Positive	True Positive (TP)	False Negative (FN)	P
Actual Negative	False Positive (FP)	True Negative (TN)	N
Total	P'	N'	Total

Key Performance Metrics

Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Proportion of correct predictions. Simple but **misleading with imbalanced data**.

Precision

$$Precision = \frac{TP}{TP + FP}$$

Of all positive predictions, how many are correct? Critical when false positives are costly.

Recall (Sensitivity)

$$Recall = \frac{TP}{TP + FN}$$

Of all actual positives, how many did we find? Critical when false negatives are costly.

F1-Score

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Harmonic mean balancing precision and recall. Better for imbalanced datasets than accuracy.

Critical Insight: A model predicting "not spam" for all emails achieves 95% accuracy if only 5% are spam—but catches zero spam! Always examine precision, recall, and F1-score alongside accuracy, especially with imbalanced classes. The choice of metric depends on your application's cost structure for different error types.