

# Machine Learning Methods for Supervised Single-Label and Multi-Label Classification

A Foundational Workshop for Education & Social Science Researchers

Yale Quan

University of Washington; Measurement & Statistics  
University of Southern Mississippi; Research Evaluation Statistics & Assessments (RESA)  
[yalequan@uw.edu](mailto:yalequan@uw.edu)

February 11, 2026

# Workshop Roadmap

- 1 What is Machine Learning?
- 2 The Big Picture
- 3 Defining the Task: The Labels
- 4 Measuring Success
- 5 The Algorithms

- 6 When You Don't Have Labels:  
Unsupervised Learning
- 7 When You Don't Have Labels:  
Unsupervised Learning
- 8 Practical Tips for Research

# What is Machine Learning?

## Traditional Programming:

Rules + Data → Answers

## Machine Learning:

Data + Answers → Rules

Instead of telling the computer *how* to classify, we show it *examples* and let it learn the patterns.

# The Big Picture: Supervised vs. Unsupervised

## Supervised Learning

- **Input:** Features ( $X$ ) + Labels ( $Y$ )
- **Goal:** Learn mapping  $f(X) \rightarrow Y$
- **Example:** Predicting student pass/fail
- **When:** You have labeled training data

## Unsupervised Learning

- **Input:** Features only ( $X$ )
- **Goal:** Discover hidden structure
- **Example:** Grouping students by behavior
- **When:** No labels available

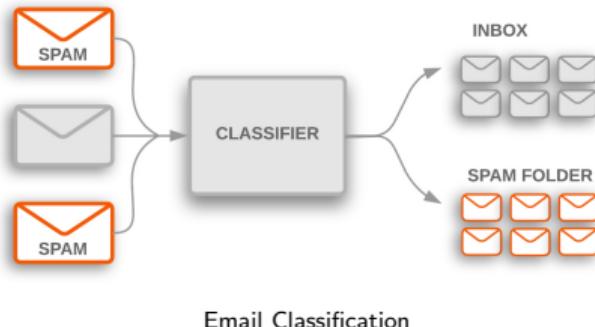
**Key Question:** Do you know the answer you're looking for?

# Why Classification Matters

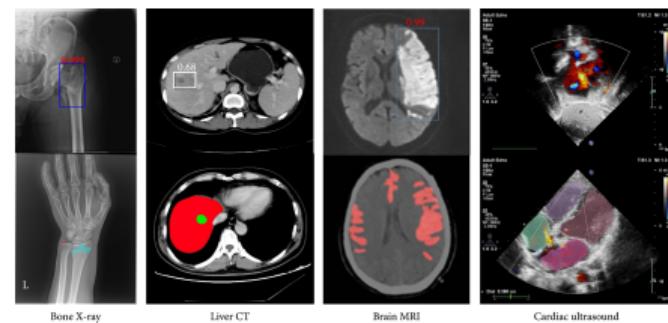
Classification helps us make sense of data by predicting known categories or discovering hidden ones.

## Real-World Examples

- **Spam Detection:** Classifying emails as "Spam" vs. "Not Spam".
- **Healthcare:** Diagnosing disease from medical images.
- **Education:** Identifying at-risk students for early intervention.



Email Classification



Medical Diagnosis

# Labels Define the Task

The structure of your target variable ( $Y$ ) determines the model you need.

## Binary Classification

**One label, two classes (0 or 1)**

Example: Pass vs. Fail, Spam vs. Not Spam

## Multi-Class Classification

**One label, 3+ mutually exclusive classes**

Example: Grades (A, B, C, D, F), Iris Species (Setosa, Versicolor, Virginica)

## Multi-Label Classification

**Multiple labels simultaneously**

Example: Movie genres (Comedy **AND** Action), Image tags (Cow **AND** Barn)

*Multi-label is fundamentally different and requires special handling!*

# Visualizing Multi-Label Classification

Consider an image classification task. A single photo can contain:

- A Cow
- A Barn
- Grass
- Sky

**Key Concept:** These are not mutually exclusive. We need a model that can predict:

$$P(\text{Cow}) > 0.5 \quad \text{AND} \quad P(\text{Barn}) > 0.5$$



*Multiple labels can apply simultaneously*

# From Problem to Evaluation

**Different problems need different metrics:**

**Binary/Multi-Class:** Single prediction per observation  
→ Confusion matrix, accuracy, precision, recall

**Multi-Label:** Multiple predictions per observation  
→ Hamming loss, subset accuracy, F1 per label

**Today's focus:** Binary and multi-class metrics  
(Multi-label evaluation is covered in the Colab session)

# The Confusion Matrix

A  $2 \times 2$  table for binary classification showing predictions vs. actual values:

		Predicted	
		Negative	Positive
Actual	Negative	TN	FP
	Positive	FN	TP

- **True Positive (TP):** Correctly identified positive cases
- **True Negative (TN):** Correctly identified negative cases
- **False Positive (FP):** Type I error (false alarm)
- **False Negative (FN):** Type II error (missed case)

# Key Evaluation Metrics

Metric	Formula
<b>Accuracy</b>	$(TP + TN) / (TP + TN + FP + FN)$
<b>Precision</b>	$TP / (TP + FP)$
<b>Recall</b>	$TP / (TP + FN)$
<b>F1 Score</b>	$2 \times [(Precision \times Recall) / (Precision + Recall)]$

# Choosing the Right Metric: Examples

## Scenario 1: Email Spam Detection (99% not spam)

- Accuracy can be misleading (always predicting "not spam" = 99% accurate!)
- Use **Precision** (avoid flagging legitimate emails) and **Recall** (catch actual spam)

## Scenario 2: Cancer Screening

- Prioritize **Recall** (don't miss any cases)
- Accept lower precision (false positives can be resolved with follow-up tests)

## Scenario 3: Student Pass/Fail (balanced classes)

- **Accuracy** is fine
- Check **F1 Score** to ensure both classes perform well

# Logistic Regression (The Baseline)

**Type:** Supervised, Linear Classifier.

## Mechanism

Estimates probability of class membership using the sigmoid function:

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-z}} \quad \text{where} \quad z = \beta_0 + \beta_1 x_1 + \dots$$

- **Decision Boundary:** Linear (draws a line/plane to separate groups).
- **Pros:** Fast and interpretable.
- **Cons:** Struggles with complex, non-linear boundaries.

# Visualizing Logistic Regression

## The Sigmoid Function:

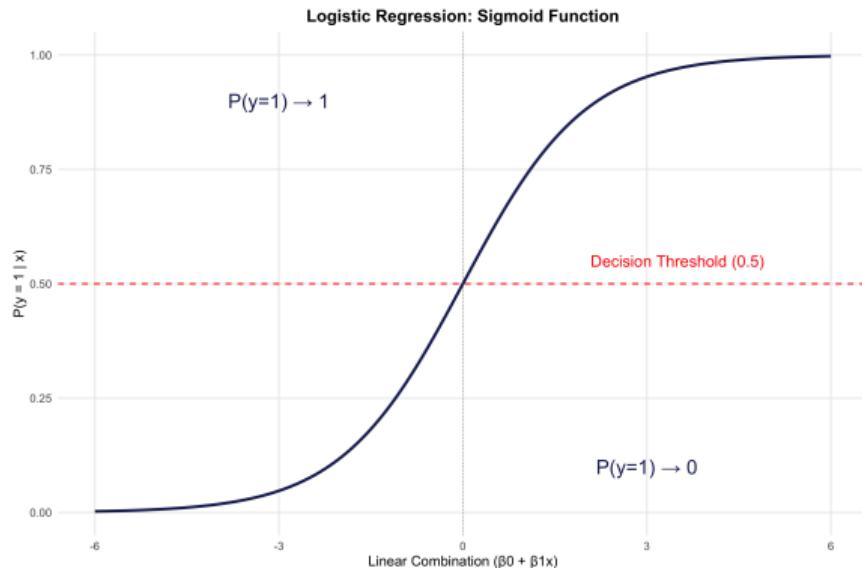
$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

## Key Properties:

- Output: Probability (0 to 1)
- S-shaped curve
- Decision boundary at 0.5

## Interpretation:

- $P > 0.5 \rightarrow$  Predict class 1
- $P < 0.5 \rightarrow$  Predict class 0



# Multinomial Logistic Regression

**Use Case:** Multi-Class Classification ( $> 2$  classes).

## The Softmax Function

Instead of a single probability, we calculate probabilities for all classes that sum to 1.

$$\sum P(y = k|\mathbf{x}) = 1.0$$

- The model estimates one set of coefficients per class (relative to a reference).
- Decision Rule: Choose the class with the highest probability ( $\arg \max P$ ).

# Visualizing Multinomial Logistic Regression

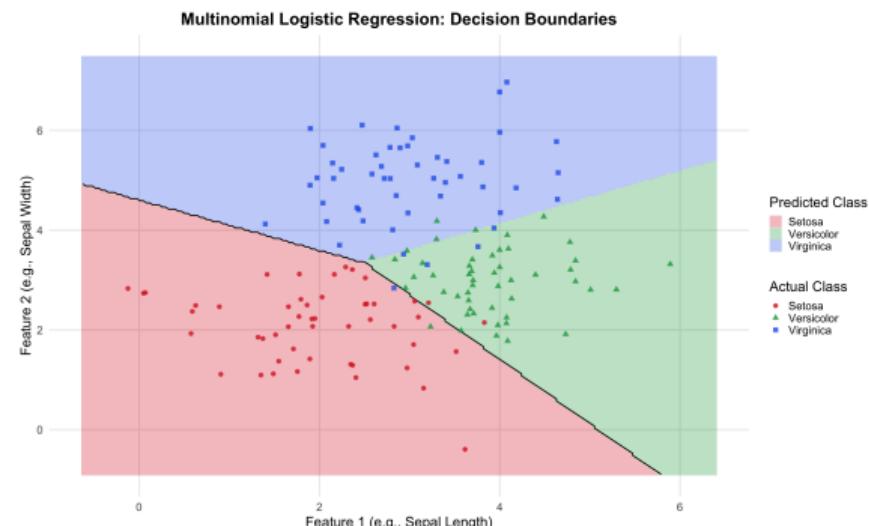
## Multiple Decision Boundaries

Unlike binary classification, we now need to separate **3+ classes**.

### Example: Iris Species

- Setosa (red)
- Versicolor (green)
- Virginica (blue)

Each region represents where one class has the **highest probability**.



# Logistic Regression: When and Why?

## Use logistic regression when:

- You need to **interpret** results (which features matter?)
- You have **limited data** (works with small samples)
- Features have **linear relationships** with the log-odds
- You need **fast training and prediction**

## Real-world applications:

- Credit scoring (explain why someone was denied)
- Medical diagnosis (interpretable risk factors)
- Social science research (theory testing)

*Start here as your baseline model!*

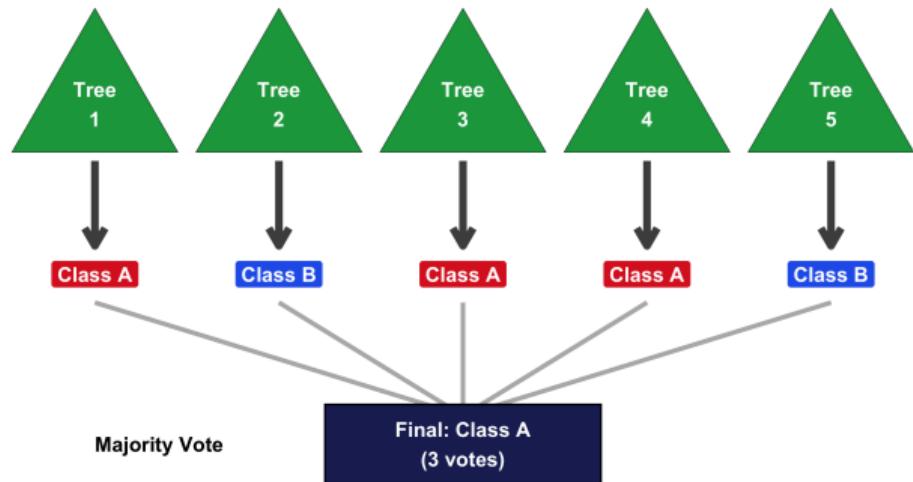
# Random Forest (Ensemble Method)

**Concept:** Combine multiple "weak" decision trees to create one "strong" predictor.

**How it works:**

- ① Randomly sample observations (Bootstrap).
- ② Train a tree, but randomly select a subset of features at each split.
- ③ Aggregate results (Majority Vote).

**Pros:** Handles non-linear relationships well; very stable.



# Random Forest vs. Logistic Regression

Characteristic	Logistic Regression	Random Forest
Interpretability	High	Low
Handles non-linearity	No	Yes
Feature interactions	Manual	Automatic
Training time	Fast	Slower
Overfitting risk	Low	Medium (with tuning)
Works with missing data	No	Yes (under specific conditions)
Typical accuracy	Good	Better

**Rule of thumb:** Start with logistic regression for interpretability, move to Random Forest for better predictive performance.

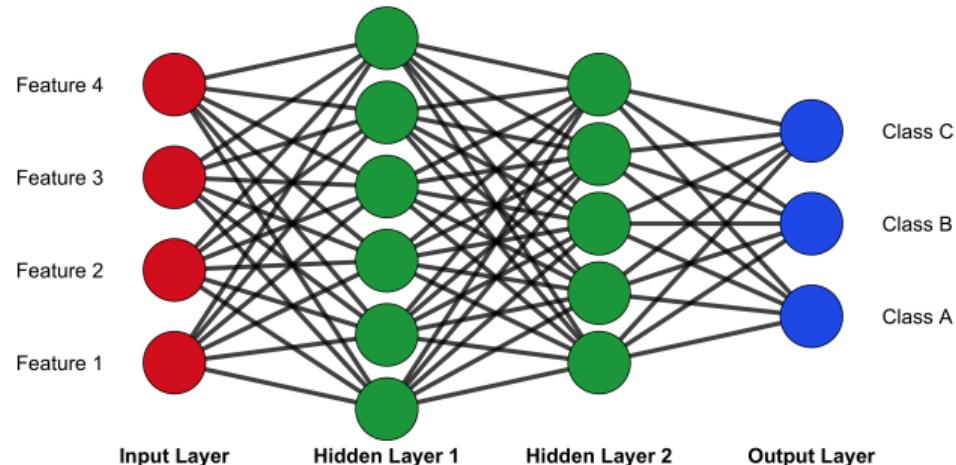
# Neural Networks (Deep Learning)

## Structure:

- Input Layer ( $X$ )
- Hidden Layers (Learn shared patterns)
- Output Layer ( $Y$ )

## Why use them?

- Captures highly complex, non-linear patterns.
- Can model correlated labels (e.g., if "barn" is present, "grass" is likely present).



# Neural Networks for Binary Classification

## The Setup

- **Output Structure:** A single output neuron.
- **Activation Function:** Sigmoid ( $\sigma$ ).
- **Goal:** Predict probability of the positive class ( $P(y = 1|X)$ ).

## The Mechanism

- The network maps the final output into a range of  $[0, 1]$ .
- **Formula:**  $\hat{y} = \frac{1}{1+e^{-z}}$

## Decision Rule

- We apply a **threshold** (usually 0.5) to classify.
- If  $P(y = 1) \geq 0.5 \rightarrow$  Predict Class 1
- If  $P(y = 1) < 0.5 \rightarrow$  Predict Class 0

## The "Logistic" Connection

This is essentially **Logistic Regression**, but the "features" fed into the final layer are learned by the hidden layers rather than being raw inputs.

# Neural Networks for Multi-Class and Multi-Label Tasks

The key difference is in the **Output Layer**.

## Multi-Class Classification

- Uses **Softmax** activation
- Probabilities **compete** with each other
- All probabilities sum to 1
- $\sum_{k=1}^K P(y = k) = 1.0$

### Example:

- $P(\text{Setosa}) = 0.7$
- $P(\text{Versicolor}) = 0.2$
- $P(\text{Virginica}) = 0.1$

*Choose one class with highest probability*

## Multi-Label Classification

- Uses independent **Sigmoids**
- Each label predicted **independently**
- Probabilities don't need to sum to 1
- Each  $P(\text{label}) \in [0, 1]$

### Example:

- $P(\text{Cow}) = 0.9$
- $P(\text{Barn}) = 0.8$
- $P(\text{Sky}) = 0.1$

*Multiple labels can be true simultaneously*

# When Do You NEED Neural Networks?

**Neural networks are usually not needed UNLESS you have:**

- ① **Unstructured data:** Images, text, audio, video
- ② **Very large datasets:** 10,000+ and/or high-dimensional
- ③ **Complex multi-label tasks:** Correlated labels
- ④ **Non-linear feature interactions:** That simpler models can't capture

**For education/social science research with tabular data:**

- Random Forest usually performs as well or better
- Much easier to tune and interpret
- Faster to train on small-medium datasets

*Don't use neural networks just because they're trendy!*

# Shifting Gears: Unsupervised Learning

**Everything so far assumed labeled data...**

But what if you:

- Don't know what categories exist?
- Want to discover natural groupings?
- Have no labels (too expensive/time-consuming to create)?

**Enter: Clustering algorithms**

Common research questions:

- Are there distinct types of learners in my classroom?
- Do survey responses cluster into meaningful groups?
- Can I identify subtypes of a phenomenon?

# When We Don't Have Labels

**Goal:** Develop a model that reveals hidden structures or patterns.

- We don't know the categories in advance.
- We want to see what groups "look like".

**Common Algorithms:**

- K-Means Clustering.
- Hierarchical Clustering.
- Gaussian Mixture Models.

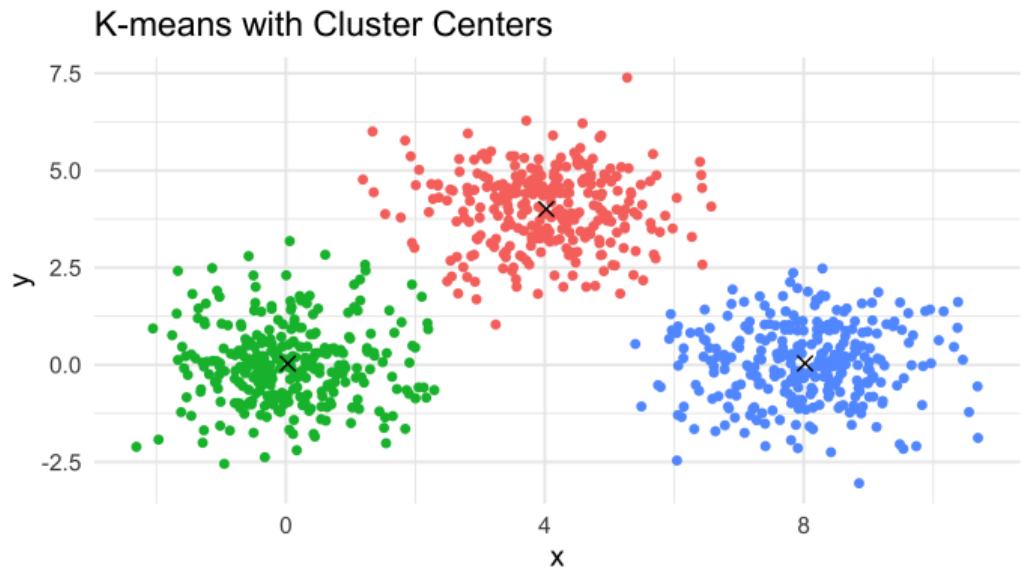
# K-Means Clustering

## Algorithm:

- ① Specify  $K$  (number of groups)
- ② Randomly place  $K$  centroids
- ③ Assign each point to nearest centroid
- ④ Update centroids to mean of assigned points
- ⑤ Repeat steps 3-4 until convergence

**Pros:** Fast, scalable, simple

**Cons:** Must choose  $K$ ; sensitive to outliers; assumes spherical clusters; Sensitive to scale (features must be standardized)



# Beyond K-Means: Other Clustering Methods

## Hierarchical Clustering

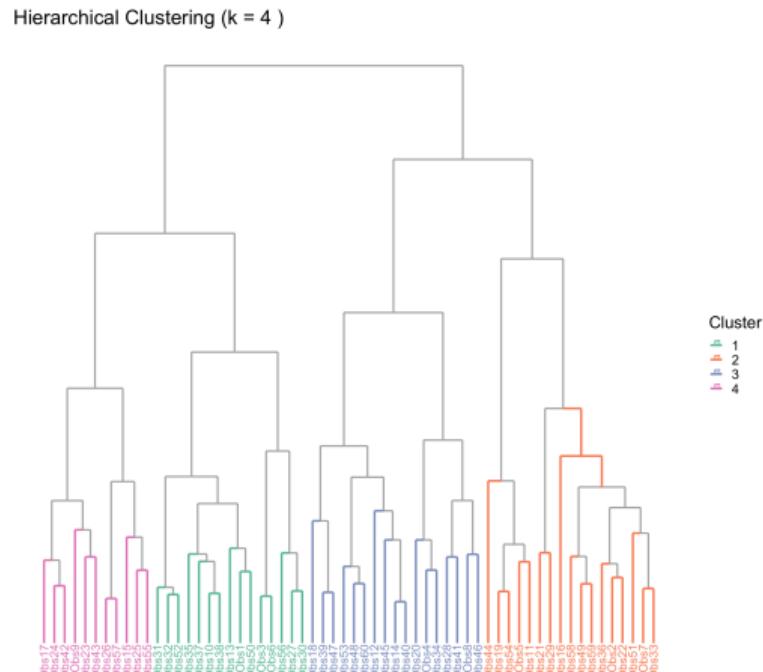
- Creates a tree (dendrogram) of nested clusters
- Can cut tree at different levels

## DBSCAN (Density-Based)

- Finds arbitrarily shaped clusters
- Identifies outliers

## Gaussian Mixture Models

- Probabilistic clustering (soft assignments)
- Models clusters as overlapping distributions



# Which Model Should You Use?

Scenario	Model	Data Size	Interpretable?
Binary/multi-class, need explanation	Logistic Reg.	Small-Large	Yes
Tabular, high accuracy needed	Random Forest	Medium-Large	No
Images, text, complex multi-label	Neural Network	Large	No
No labels, explore structure	K-Means	Any	N/A
Non-spherical clusters	DBSCAN	Medium	N/A

## Recommended workflow:

- ① Start simple (Logistic Regression)
- ② Try Random Forest for better performance
- ③ Only use Neural Networks if you have sufficient data/complexity

# Practical Tips for Research

## Before you start:

- Define your research question clearly
- Ensure you have enough data (rule of thumb: 10+ samples per feature)
- Split data: Train (60%), Validation (20%), Test (20%)

## During model building:

- Always establish a baseline (e.g., logistic regression)
- Use cross-validation to avoid overfitting
- Check for class imbalance

## Common pitfalls to avoid:

- Testing on training data (data leakage!)
- Ignoring class imbalance
- Over-interpreting small accuracy improvements

# Feature Scaling and Standardization

Many machine learning algorithms are **sensitive to the scale of your features**.

## Why this matters:

- Variables measured on larger scales can dominate the model
- Distance-based methods are especially affected
- Coefficients and regularization depend on scale

## Algorithms that REQUIRE scaling:

- K-Means clustering
- K-Nearest Neighbors (KNN)
- Support Vector Machines (SVM)
- Logistic Regression (with regularization)
- Neural Networks

## Common approaches:

- **Standardization (Z-score):**  $(x - \mu)/\sigma$
- **Min-Max Scaling:** Rescale to  $[0, 1]$

## What's Next:

### Google Colab Session

- Implement logistic regression, random forest, and neural networks
- Compare performance on real datasets
- Explore multi-label classification
- Try K-means clustering

### Resources:

- Slides & code: [your link]

### Questions after the workshop?

email: [yalequan@uw.edu](mailto:yalequan@uw.edu)