Session – 19

# ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

## Data Handling: Scaling, Encoding, Variable Types, Feature Selection & Model Selection

## AIM OF THE SESSION

To familiarize students with the basic concept of Data Handling, Feature Selection and Model Selection

## INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. Explain Data Handling, Feature Selection and Model Selection.

## LEARNING OUTCOMES

At the end of this session, you should be able to:

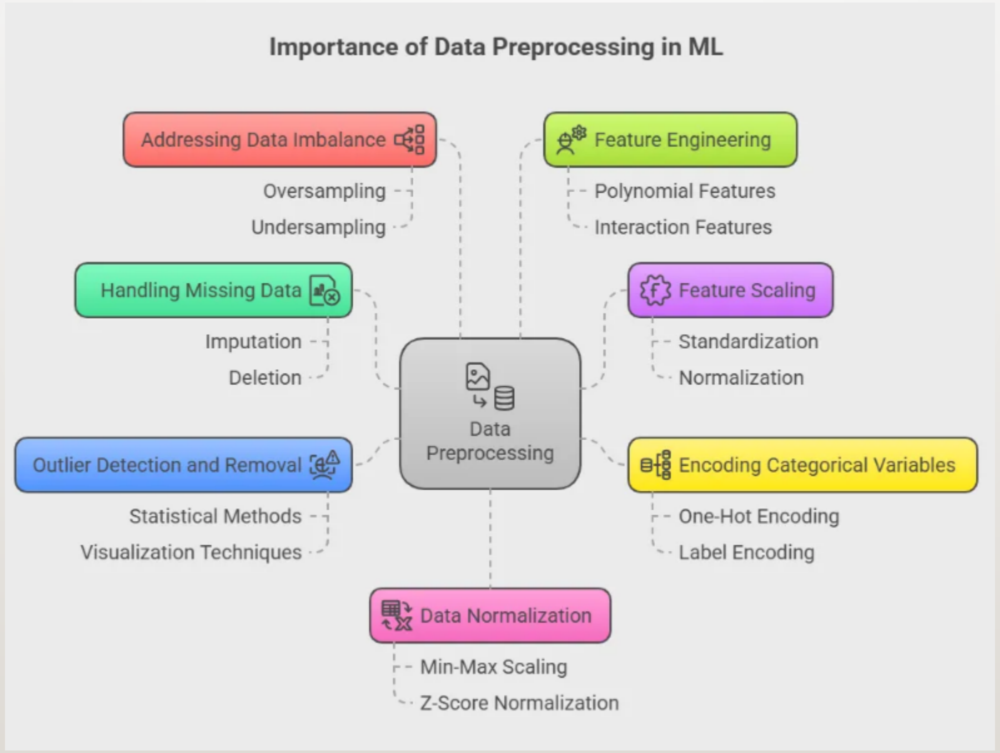
1. Define the concepts of Data Handling, Feature Selection and Model Selection.

## Outline

- The Importance of Data Preprocessing
- Variable Types: Knowing Your Data (Numerical vs. Categorical)
- Feature Scaling: Standardization vs. Normalization
- Encoding Categorical Variables: One-Hot vs. Label vs. Embeddings
- The Curse of Dimensionality & Benefits of Feature Selection
- Feature Selection vs. Dimensionality Reduction: What's the difference?
- Filter Methods: Statistical Scores (Fisher Score, Mutual Information)
- Wrapper Methods: Intelligent Search (Recursive Feature Elimination)
- Embedded Methods: Model-Based Selection (Lasso, Tree Importance)
- Model Selection: The No Free Lunch Theorem & Introduction to Aggregation

# The Importance of Data Preprocessing

▪ **Raw data is rarely model-ready.** Preprocessing is the essential step of cleaning and transforming raw data into a format that machine learning algorithms can understand effectively.



# The Importance of Data Preprocessing

## Why Bother?

✓**Algorithm Requirements:** Many algorithms (e.g., SVM, K-Means, Gradient Descent-based models) require scaled data to perform correctly.

✓**Performance:** Improves model convergence speed and accuracy.

✓**Fairness:** Prevents features with larger scales (like salary) from dominating those with smaller scales (like age).

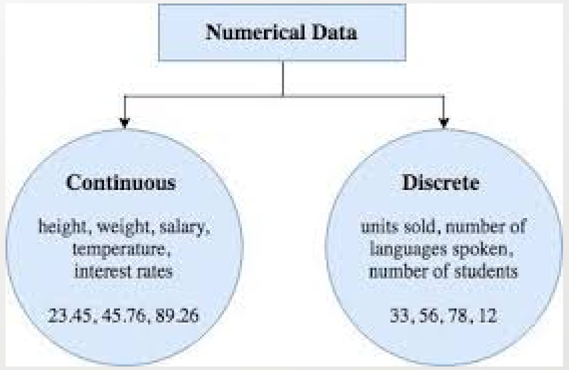
✓**Compatibility:** Algorithms only work with numerical values.

# Know Your Data: Variable Types

## Categorical vs. Numerical Data

### Categorical (Qualitative)

- Represents groups or categories.
- **Nominal:** No order (e.g., Country, Color, Product Type).
- **Ordinal:** Has a meaningful order (e.g., Education Level, Satisfaction Rating (Low, Med, High)).


Numerical Data
Continuous
height, weight, salary, temperature, interest rates
23.45, 45.76, 89.26
Discrete
units sold, number of languages spoken, number of students
33, 56, 78, 12

### Numerical (Quantitative)

- Represents measurable quantities.
- **Discrete:** Countable integers (e.g., Number of Children, Page Views).
- **Continuous:** Infinite possible values (e.g., Height, Weight, Temperature).

## Problem: Features on Different Scales

### The Dominating Feature Problem

- **Algorithms using distance calculations are highly sensitive to feature scales.**

**Example: K-Nearest Neighbors (KNN)**

Feature 1: Age (Range: 20 - 80 years)

Feature 2: Salary (Range: 40,000 - 120,000 USD)

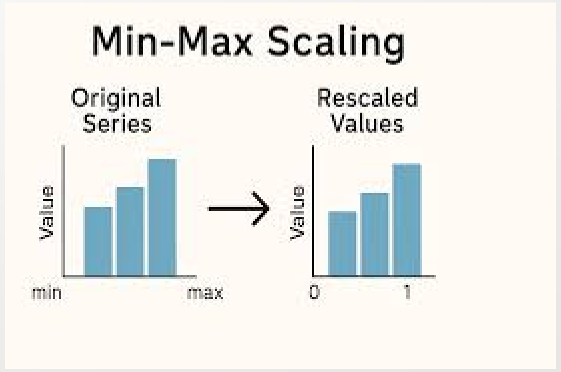
- **The distance between points will be dominated by Salary,** making Age almost irrelevant to the model's calculations.

**Solution: Feature Scaling**

# Solution: Feature Scaling - Normalization (Min-Max Scaling)

## Normalization (Min-Max Scaling)

- Rescales features to a fixed range, usually **[0, 1]**.



## Formula:

**X_new = (X - X_min) / (X_max - X_min)**

## Pros:

- Simple to implement.
- Preserves the original distribution.
- Good for algorithms that don't assume a distribution (e.g., KNN, Neural Networks).

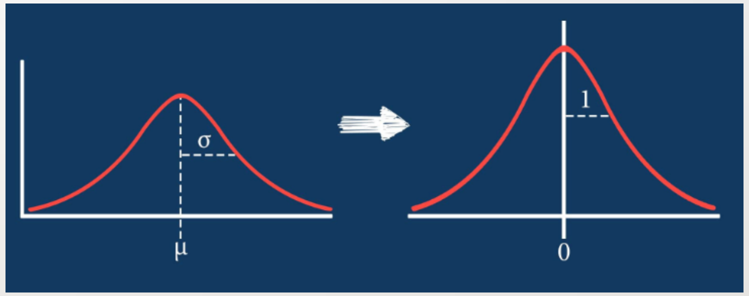
## Cons:

- **Sensitive to outliers.** A single extreme outlier can compress the rest of the data.
- **Use when:** You know the data is bounded or you have no major outliers.

# Solution: Feature Scaling - Standardization (Z-Score Normalization)

## Standardization (Z-Score Normalization)

- Rescales data to have a **mean of 0** and a **standard deviation of 1**.



## Formula:

**X_new = (X - μ) / σ**

## Pros:

**Less sensitive to outliers** than Min-Max scaling.

- Works well for algorithms that assume a Gaussian (normal) distribution (e.g., Linear Regression, Logistic Regression).

## Cons:

- Does not bound values to a specific range.
- **Use when:** The data contains outliers or when the algorithm assumes a normal distribution.

## Encoding Categorical Variables: The Problem

### Machines Don't Speak English

- Machine learning models are mathematical constructs. They require **numerical input**.

### We cannot do this:

**model.fit([["UK"], ["USA"], ["France"]])**

**We must convert text labels to numbers. But how?**

**Incorrect Approach: Label Encoding (0, 1, 2)**

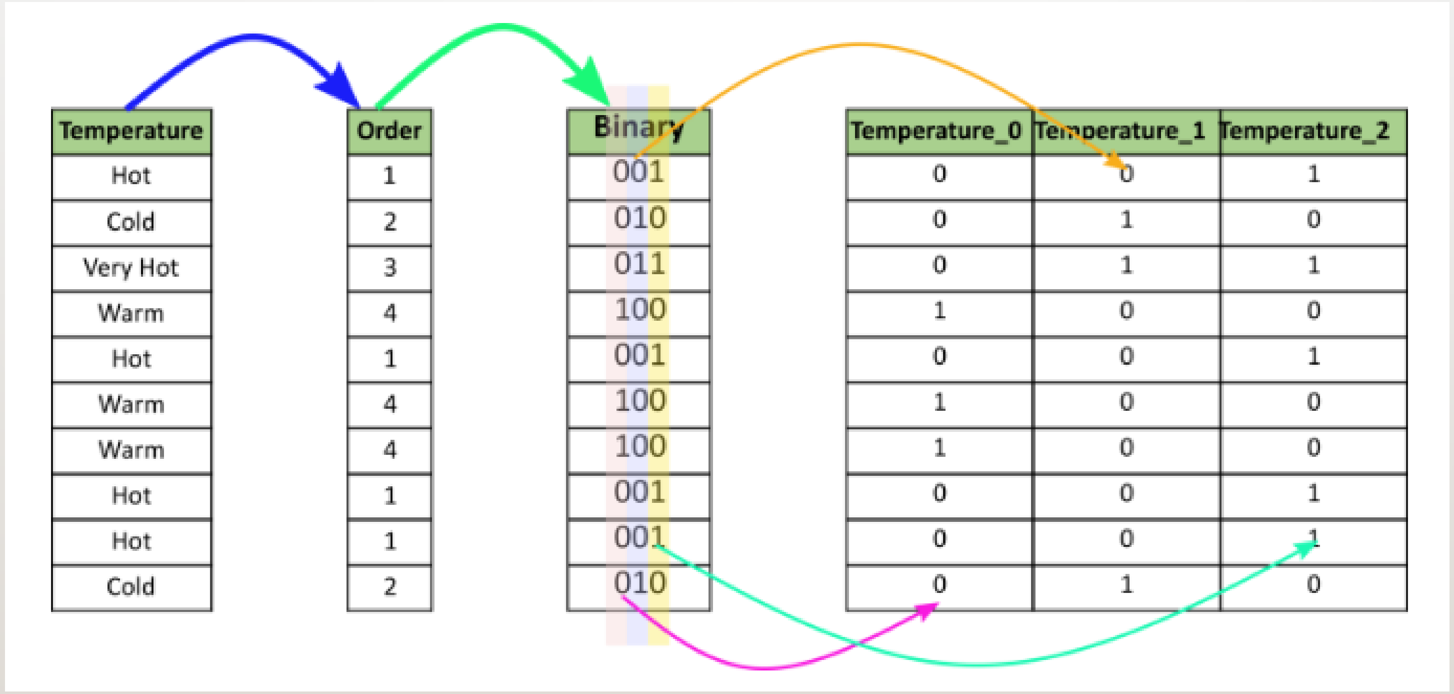
UK=0, USA=1, France=2

Problem: The model might incorrectly assume an **order** (France > USA > UK), which doesn't exist for nominal data.

# Encoding Solution #1: One-Hot Encoding

## One-Hot Encoding

Creates **new binary (dummy) columns** for each category.



| Temperature |
|---|
| Hot |
| Cold |
| Very Hot |
| Warm |
| Hot |
| Warm |
| Warm |
| Hot |
| Hot |
| Cold |

| Order |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| 1 |
| 4 |
| 4 |
| 1 |
| 1 |
| 2 |

| Binary |
|---|
| 001 |
| 010 |
| 011 |
| 100 |
| 001 |
| 100 |
| 100 |
| 001 |
| 001 |
| 010 |

| Temperature_0 | Temperature_1 | Temperature_2 |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |

## Encoding Solution #1: One-Hot Encoding

**Pros:**

- Prevents false ordinal relationships.
- Works well with nominal data.

**Cons:**

- **Curse of Dimensionality:** If a category has many unique values (e.g., Zip Code), it creates many new columns, making the dataset sparse.

## Encoding Solution #2: Label Encoding

**Label Encoding**

- **Only use Label Encoding for Ordinal Variables** where the order is meaningful.

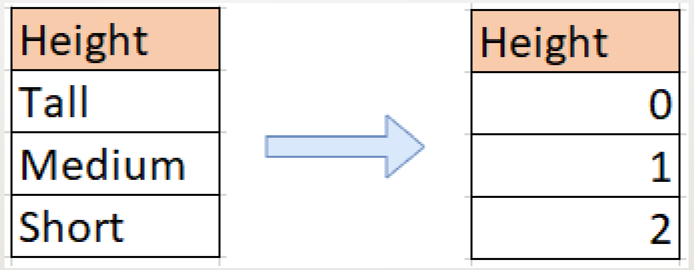
**Example: Education Level**

High School = 0

Bachelor's = 1

Master's = 2

PhD = 3



| Height |
|---|
| Tall |
| Medium |
| Short |

| Height |
|---|
| 0 |
| 1 |
| 2 |

- Here, the numerical order **correctly** represents the inherent order of the categories.
- **Do NOT use for nominal data.**

## Encoding Solution #3: Embeddings (Advanced)

### Embeddings for High-Cardinality Data

- An **advanced** technique that learns a meaningful, lower-dimensional representation of categories.
- Instead of a sparse one-hot vector [0,0,1,0,...,0], an embedding is a dense vector like [0.24, -0.87, 0.19].

### Key Benefit:

- Similar categories (e.g., "Dog" & "Cat") will have similar vector representations.
- Learned automatically by models like Neural Networks during training.
- Commonly used in NLP (Word2Vec) and for user/item IDs in recommendation systems.

## Typical Preprocessing Workflow

### A Standard Data Handling Pipeline

1. **Split Data:** First, split your data into Training and Test sets! **Never apply preprocessing before splitting** to avoid data leakage.
2. **Handle Missing Values:** Impute or remove missing data.
3. **Encode Categorical Variables:** Use One-Hot for nominal, Label for ordinal.
4. **Scale Numerical Features:** Apply Standardization or Normalization.
5. **Train Model:** Fit the model on the processed training data.
6. **Evaluate:** Test on the processed test data.

**Critical: Fit the scaler/encoder on the training data only**, then use it to transform both the training and test data.

## Feature Selection & Model Selection

# The Problem: The Curse of Dimensionality

## Why Do We Need to Select Features?

## Dimensionality

➢As the number of features (dimensions) grows, the volume of the space increases so fast that the available data becomes **sparse**. This sparsity makes it difficult to find meaningful patterns.

## Benefits of Feature Selection

✓**Reduces Overfitting:** Less redundant data means less opportunity to make decisions based on noise.

✓**Improves Accuracy:** Removes misleading or irrelevant features.

✓**Faster Training:** Fewer features reduce computational cost.

✓**Simpler Models:** Models are easier to interpret and explain.

## Feature Selection vs. Dimensionality Reduction

### Two Strategies for Simpler Models

| Aspect | Feature Selection | Dimensionality Reduction (e.g., PCA) |
|---|---|---|
| **Output** | **Subset of original features** | **New, transformed features** |
| **Interpretability** | **High.** The original features and their meaning are preserved | **Low.** New features (components) are hard to interpret. |
| **Goal** | Identify the most relevant existing variables. | Capture the most variance in the data, regardless of feature meaning. |

## Filter Methods: The Statistical Screening

### Filter Methods

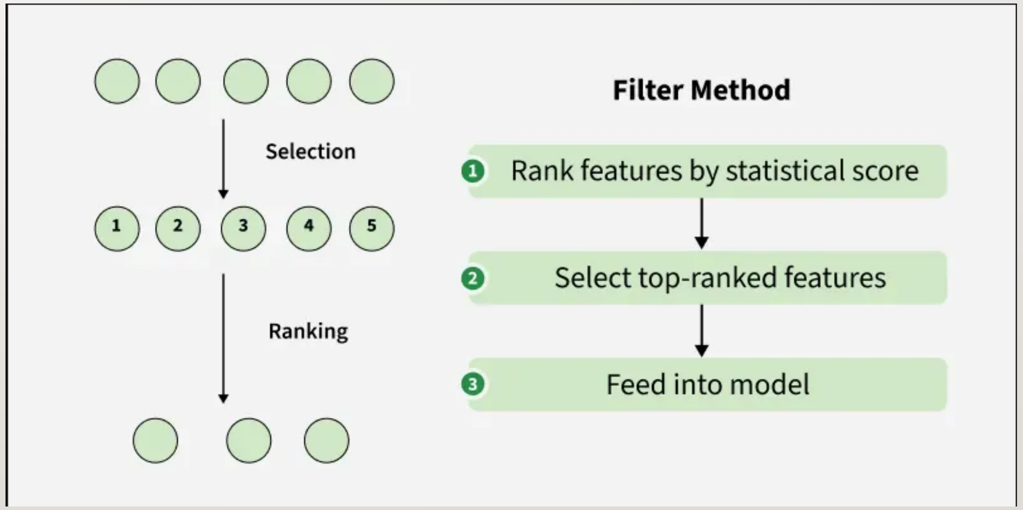
➢Features are selected based on their **statistical scores** in relation to the target variable. It's a pre-processing step, independent of any machine learning model.

### How it works:

- oCalculate a score for each feature.
- oRank the features by their score.
- oSelect the top *k* features.

**Pros:** Fast, scalable, model-agnostic.

**Cons:** Ignores feature interactions and the model itself.



Filter Method
Selection
1
2
3
4
5
Ranking
1 Rank features by statistical score
2 Select top-ranked features
3 Feed into model

## Filter Method: Fisher Score

### Fisher Score (Filter Method)

➢ Measures the **separation** between the classes for a given feature.

### A good feature will have:

- **Low variance** *within* each class (values are similar).
- **High variance** *between* different classes (values are different).

### Formula (Conceptual):

**Fisher_Score = (Between-Class Variance) / (Within-Class Variance)**

- **A higher score means the feature is better at separating the classes.**

Use Case: Excellent for **classification** problems.

# Filter Method: Mutual Information (MI)

## Mutual Information (Filter Method)

➢Measures the **amount of information** one can obtain about the target variable (Y) by observing the feature (X). It's based on information theory (entropy).

**Idea:** How much does knowing the value of feature X reduce our uncertainty about the class Y?

**Formula (Conceptual):**

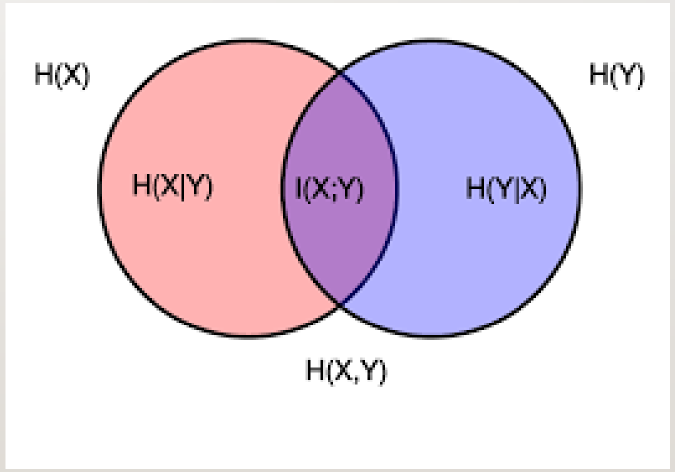
**MI(X; Y) = H(X) + H(Y) - H(X, Y)**

where H is entropy (a measure of uncertainty).

**MI = 0:** Feature X and target Y are independent.

**MI > 0:** Knowing X gives us information about Y.

**Use Case:** Works for both classification and regression.



# Wrapper Methods: The Intelligent Search

## Wrapper Methods

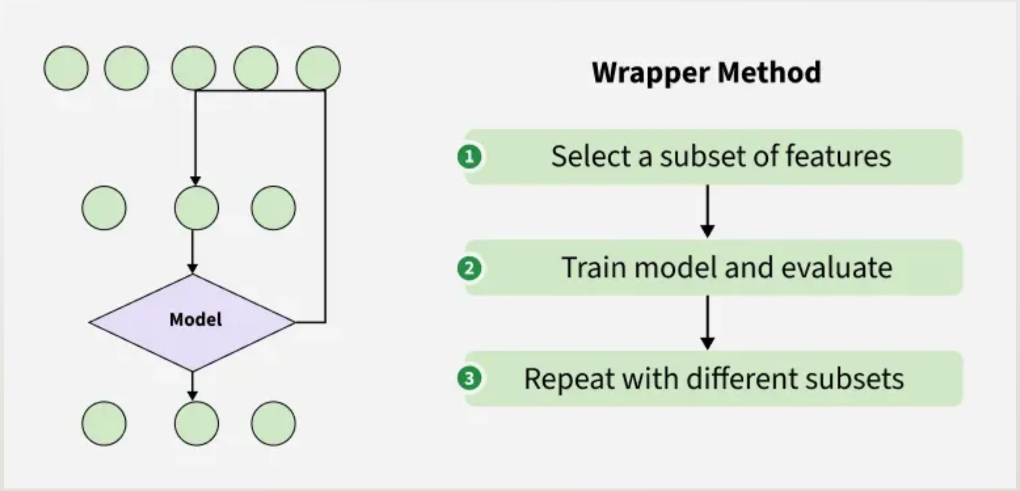
Use a specific machine learning model to evaluate the quality of a **subset of features**. They "wrap" around a model.

## How it works:

1. Search for a subset of features.
2. Train a model on that subset.
3. Evaluate the model's performance (e.g., accuracy).
4. Repeat until the best subset is found.

**Pros:** Model-specific, can find interacting features.

**Cons:** Computationally very expensive, high risk of overfitting.


Wrapper Method
Model
1 Select a subset of features
2 Train model and evaluate
3 Repeat with different subsets

# Wrapper Method: Recursive Feature Elimination (RFE)

## Recursive Feature Elimination (RFE) (Wrapper Method)

- A popular wrapper method that works by **recursively removing the least important features**.

**Steps:**

1. Train a model on all features.
2. Get the feature importance (e.g., from coefficients in linear models).
3. Discard the least important feature(s).
4. Retrain the model on the remaining features.
5. Repeat steps 2-4 until the desired number of features is reached.

**The result is a ranking of features and an optimal subset.**

## Embedded Methods: The Best of Both Worlds?

### Embedded Methods

➢Feature selection is **built into the model's training process**. The model itself performs feature selection as it learns.

### Pros:

- Model-specific like Wrappers (so they are effective).
- Faster than Wrappers (no need to re-train on multiple subsets).

### Examples:

- **Lasso (L1) Regression:** Adds a penalty that forces some feature coefficients to exactly zero, effectively removing them.
- **Tree-Based Algorithms** (Random Forest, XGBoost): Provide natural feature importance scores based on how much a feature decreases impurity across all trees.

## Model Selection: The No Free Lunch Theorem

### Choosing the Right Algorithm

### The No Free Lunch Theorem

- "There is no single model that is best for every problem."
- The performance of a model is dependent on the specific data and problem.

### How to Choose?

**Data Size & Quality:** Neural Nets need lots of data; SVMs work well with clear margins.

**Problem Type:** Classification vs. Regression.

**Interpretability:** Does the model need to be explainable (e.g., Linear vs. Random Forest)?

**Training Time & Complexity.**

# Model Aggregation: Ensemble Methods

## Aggregation (Ensemble Learning)

Combine predictions from multiple models to improve robustness and accuracy.

## Key Ideas:

**Bagging (Bootstrap Aggregating):** Trains many models in parallel on different data subsets and averages their predictions. **Reduces variance.** (e.g., Random Forest).

**Boosting:** Trains models sequentially, where each new model tries to correct the errors of the previous one. **Reduces bias.** (e.g., AdaBoost, XGBoost).

**Stacking:** Uses a meta-model to learn how to best combine the predictions of several base models.

**Result:** Ensemble models are often among the top performers in machine learning competitions.

## Summary

- **Know Your Data:** Always identify variable types (Numerical vs. Categorical, Nominal vs. Ordinal) first.
- **Scale Numerical Features:** Use Standardization by default; use Normalization if you have bounded data and no outliers.
- **Encode Categorical Features:** Use One-Hot for nominal data, Label for ordinal data. Consider Embeddings for complex problems.
- **No Data Leakage:** Always split your data before any preprocessing step.
- **Proper data handling is not optional—it's the key to building robust and accurate models.**
- Feature selection reduces overfitting, improves accuracy, and speeds up training.
- **Filter (Fisher, MI):** Fast first pass, **Wrapper (RFE):** Accurate but slow, **Embedded (Lasso, Trees):** Often the best practical choice.
- **No Perfect Model:** Model choice depends on the problem (No Free Lunch Theorem).
- **Aggregate for Power:** Ensemble methods (Bagging, Boosting) combine weak models to create a strong one.

## References

**Books:**

1. **Book:** Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow
   **Author:** Aurélien Géron
   **Editions:** 3rd Edition (2022) is best, but 2nd is also excellent.

2. **Book:** An Introduction to Statistical Learning with Applications in R or Python
   **Authors:** Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

3. **Book:** The Elements of Statistical Learning (ESL)
   **Authors:** Trevor Hastie, Robert Tibshirani, Jerome Friedman

**Web Link:**

1. http://www.statlearning.com/

# THANK YOU