# Classification in Machine Learning

## Elements of AIML

.

# Introduction

.

**Definition**: Classification is a supervised learning technique used to categorize data into predefined classes or labels.

**Purpose**: To build a model that can assign new, unseen data points to one of these classes based on the learned patterns.

**Examples**: Email spam detection, image recognition, medical diagnos

# Key Points about Classification

.

**Labeled Data**: Classification requires labeled datasets for training.

**Predictive Modeling**: Helps in making predictions on categorical outcomes.

**Evaluation Metrics**: Commonly evaluated using accuracy, precision, recall, F1-score, and confusion matrix.

**Binary vs. Multi-Class**: Classification can be binary (two classes) or multi-class (more than two classes).

# Types of Classification Algorithms

.

**Overview**: Classification algorithms can be grouped into various types based on their working principles.

Various types are following as:

1 Logistic Regression

2. Decision Tree

3. Naive Bayes Classifier

4. K-Nearest Neighbors (KNN) Classifier

5. SVM

6 Neural Network Classifier

# Logistic Regression

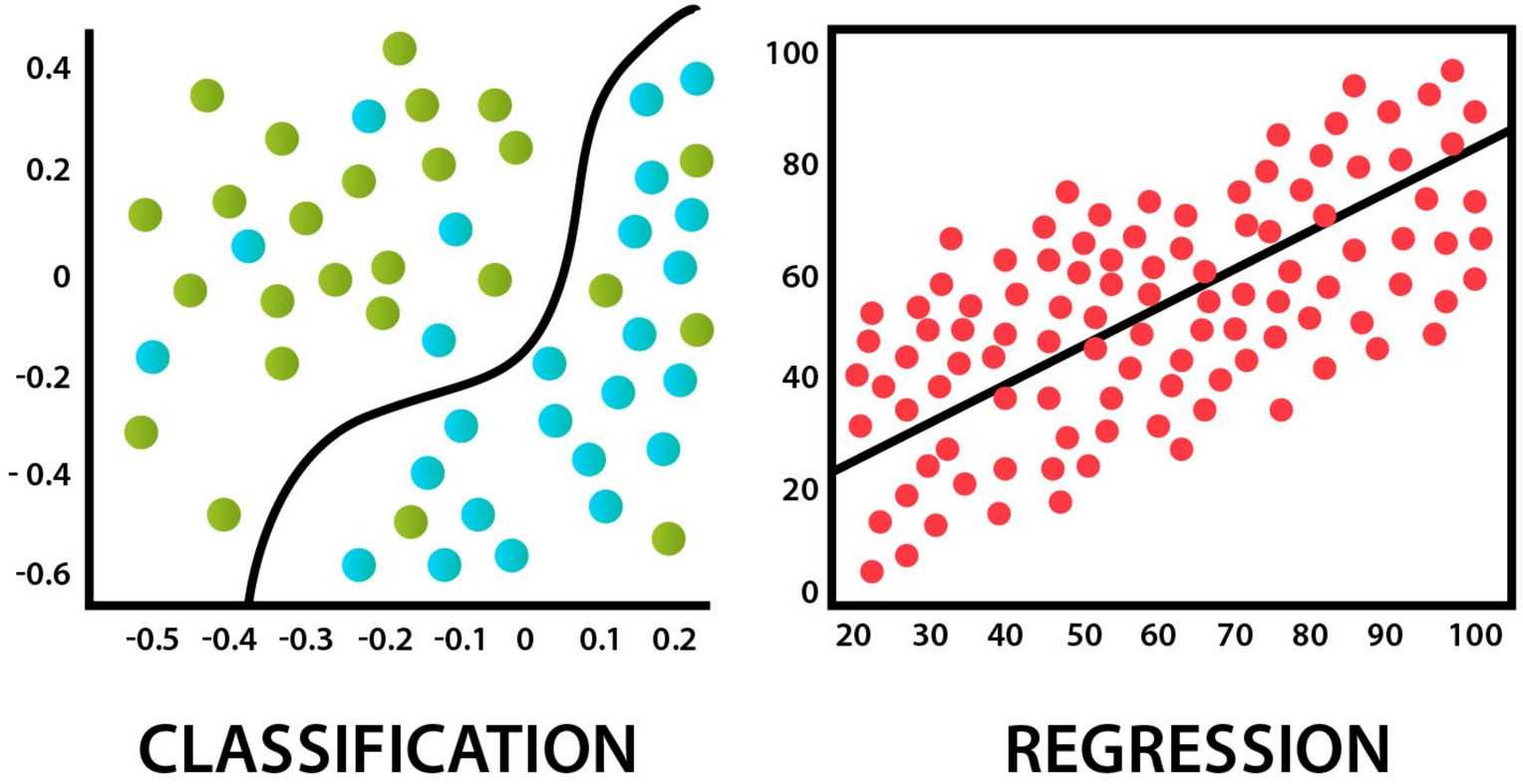
▪

- It is statistical method used in machine learning primarily for **binary classification** tasks, where the goal is to classify data points into one of two categories.
- such as “yes” or “no,” “spam” or “not spam,” etc.
- Logistic Regression predicts the probability that a given input belongs to a particular class.

# Diagram

- 

- .


0.4
0.2
0
-0.2
- 0.4
-0.6
-0.5 -0.4 -0.3 -0.2 -0.1 0 0.1 0.2
CLASSIFICATION
100
80
60
40
20
0
20 30 40 50 60 70 80 90 100
REGRESSION

# Key Points of Logistic Regression:

.

**Classification, Not Regression**: Despite its name, Logistic Regression is used for classification problems, not regression.

**Probability-Based**: It predicts the probability of a data point belonging to a particular class, usually in the range [0, 1].

**Sigmoid Function**: Logistic Regression uses a Sigmoid function to map predictions to probabilities.

**Decision Boundary**: The model decides the class based on a threshold (usually 0.5) applied to the probability.

# Mathematical Foundation of Logistic Regression

.

**Linear Component:**

**Similar to Linear Regression, Logistic Regression starts by calculating a linear combination of the input features.**

**This can be written as:**

$$z = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

**where:**

**z is the linear predictor.**

$w_0, w_1, \ldots, w_n$ **are the model weights.**

$x_1, x_2, \ldots, x_n$**are the feature values.**

# Sigmoid (Logistic) Function:

.

To map the linear predictor $_z$ to a probability, Logistic Regression uses the Sigmoid function, defined as:

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

The Sigmoid function takes any real-valued number and maps it to a value between 0 and 1, making it suitable for probability predictions.

The final predicted probability P(y=1|x) for the positive class is:

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \ldots + w_n x_n)}}$$

# Decision Rule:

- The output probability P(y=1|x) is compared with a threshold (usually 0.5).
  - If P(y=1|x) \geq 0.5 , classify as the positive class (1).
  - If P(y=1|x) < 0.5 , classify as the negative class (0).

**Binary Cross-Entropy (Log Loss)**

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)\right)$$

N : Number of data points.

y_i : Actual label (0 or 1) of the i -th data point.

p_i : Predicted probability of the positive class for the i -th data point.

# Advantages of Logistic Regression

interpretability: Coefficients  w_i  provide insights into the feature impact.

**Probability Output**: Provides a probability score, useful for more granular analysis.

**Efficient**: Computationally efficient for binary classification problems.

**Drawback:**

**Limited to Linear Boundaries**: Assumes a linear relationship between features and the log-odds of the outcome.

- **Sensitive to Outliers**: Logistic Regression can be affected by outliers in the data.

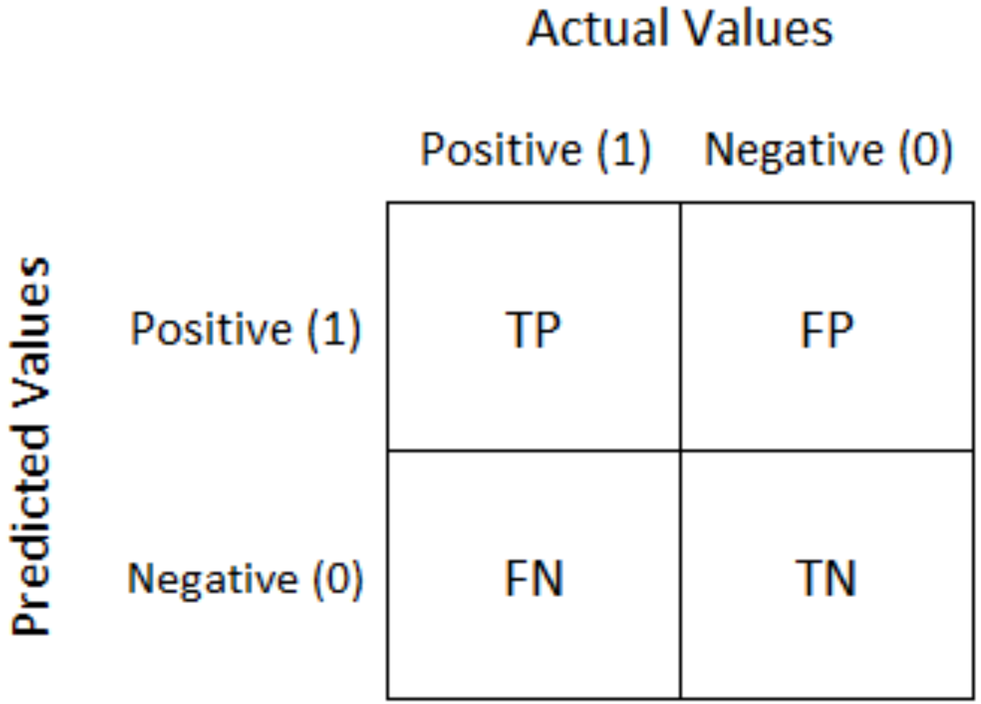
# Confusion Matrix

A **Confusion Matrix** is a tool used to evaluate the performance of a classification model. It provides insights into the types of errors the model makes, beyond just accuracy

**True Positive (TP)**: Correctly predicted positive cases.

**False Positive (FP)**: Incorrectly predicted positive cases (Type I error).

**True Negative (TN)**: Correctly predicted negative cases.




**False Negative (FN)**: Incorrectly predicted negative cases (Type II error).

# Key Metrics from Confusion Matrix

**Accuracy:**

- Formula: {Accuracy} = $\frac{TP + TN}{TP + TN + FP + FN}$
- Measures the proportion of correctly predicted cases.

**Precision (Positive Predictive Value):**

- Formula: Precision = $\frac{TP}{TP + FP}$
- Measures the accuracy of positive predictions.

# Key Metrics

.

**Recall (Sensitivity or True Positive Rate):**

Formula : $\text{Recall} = \frac{TP}{TP + FN}$

Indicates how well the model identifies positive instances.

**F1 Score:**

Formula: $\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

**Balances precision and recall, especially useful when classes are imbalanced.**

# Applications of Confusion Matrix

.

**Performance Evaluation**: Widely used in machine learning for evaluating model accuracy, especially in binary and multi-class classification.

**Healthcare**: Helps in evaluating models used for medical diagnosis to balance sensitivity (recall) and specificity.

**Spam Detection**: Analyzes model effectiveness in classifying emails as spam or not spam.

**Fraud Detection**: Useful in identifying correct and incorrect predictions in financial transaction monitoring.

# Advantages of Confusion Matrix

.

**Comprehensive Insight**: Offers a complete view of how predictions are distributed across all classes.

**Metric Calculation**: Allows calculation of multiple evaluation metrics such as accuracy, precision, recall, and F1 score.

**Error Identification**: Helps identify specific types of errors (false positives and false negatives).

# Drawbacks of Confusion Matrix

.

**Not a Performance Score Alone**: It’s a diagnostic tool, not a direct measure of model quality.

**Class Imbalance Issue**: Accuracy can be misleading for imbalanced datasets, where one class dominates.

**No Model Insights**: Provides no information on why errors occur, only that they do.

**Limited for Non-Binary Classification**: More complex to interpret for multi-class problems.

# Decision Tree

# Introduction

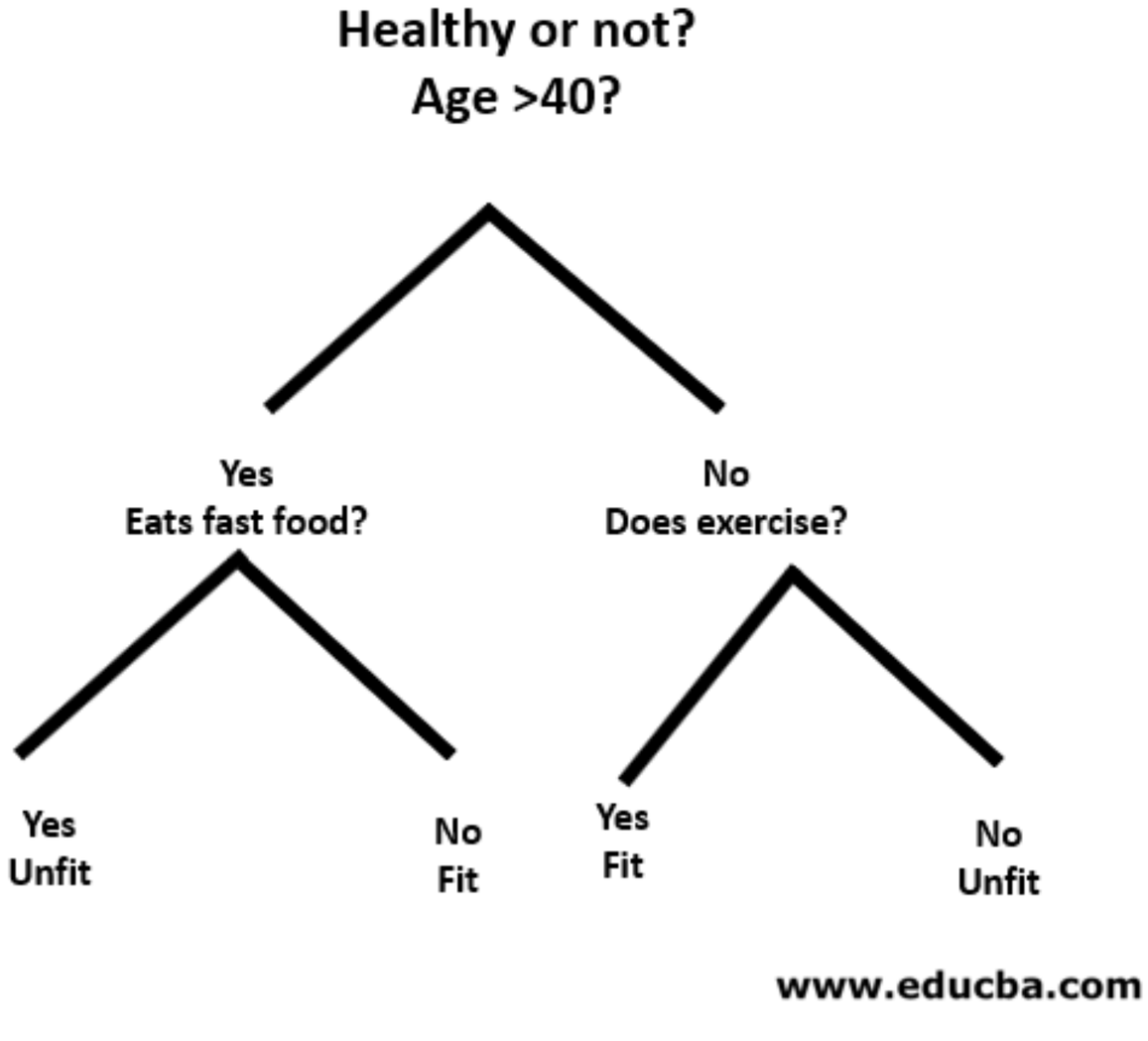
.

**Definition**: It is a powerful machine learning model used for both classification and regression tasks. It operates by recursively splitting the dataset into subsets based on the value of the features

- Splits data based on the feature that offers the best division (e.g., Gini Impurity or Information Gain).
- Continues splitting until a stopping condition (e.g., max depth or minimum samples per leaf) is reached.

# Diagram

-
- .


Healthy or not?
Age >40?
Yes
Eats fast food?
No
Does exercise?
Yes
Unfit
No
Fit
Yes
Fit
No
Unfit
www.educba.com

# Key Points of Decision Trees

▪

- I**nterpretable Structure**: Each decision path in the tree can be easily interpreted, making it useful for explainable AI.
- **Non-linear Relationships**: Captures complex, non-linear relationships between features and target outcomes.
- **Feature Importance**: Highlights which features are most important for predictions.
- **No Assumptions on Data**: Does not require assumptions about data distribution (unlike some algorithms like linear regression).
- Handles Both Numeric and Categorical Data: Flexible for a variety of data types.

# How Decision Trees Work

.

1. **Splitting Criteria**: Decision trees split data at each node based on a metric like

   - **Gini Impurity**
   - Entropy
   - Information Gain

2. **Recursive Splitting**: Continues splitting nodes into branches until a stopping criterion is met (e.g., maximum depth, minimum number of samples).

3. **Leaf Node Assignment**: When a node reaches a stopping condition, it becomes a leaf node, assigned a class label or value.

# Gini Impurity

.

- **Gini Impurity**: Measures how often a randomly chosen element would be misclassified.
  Range is 0 to 0.5

Formula: $Gini = 1 - \sum_{i=1}^{C} p_i^2$

The lower the Gini Impurity, the better the split.

It’s generally faster to compute than entropy,

# Entropy

.

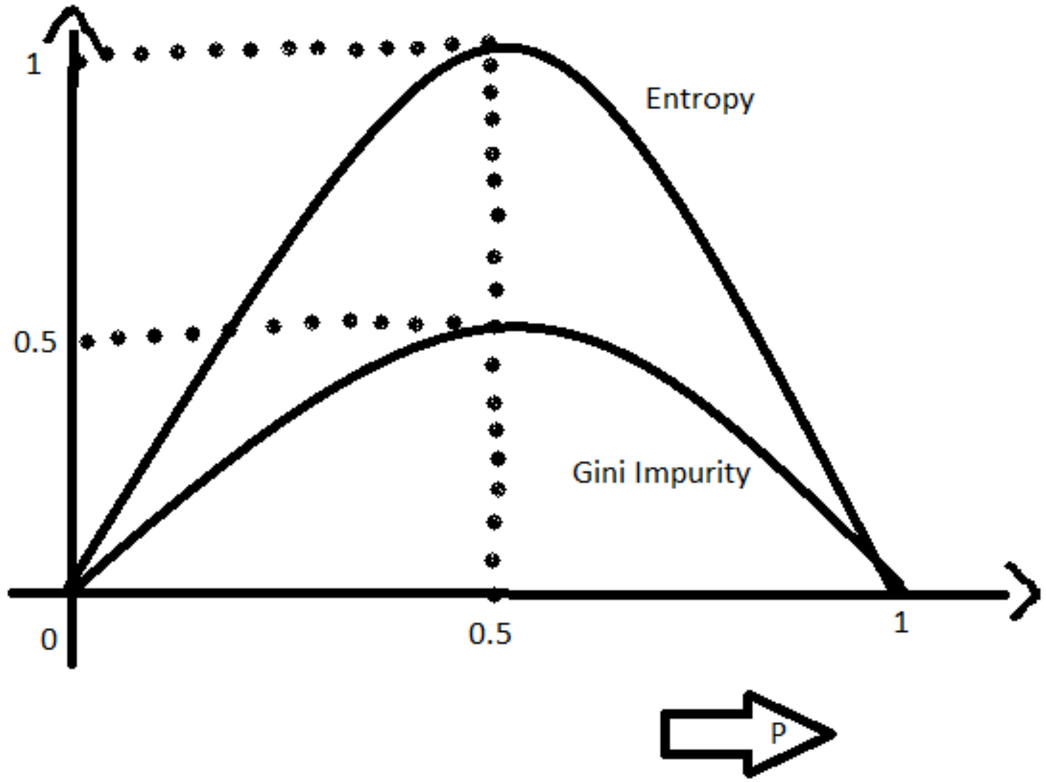
- Entropy: It measures the level of impurity or randomness in a dataset.

$$Entropy = -\sum_{i=1}^{C} p_i \cdot \log_2(p_i)$$

where p_i is the probability of a particular class i , and C is the number of classes.

Range: Entropy values range from 0 (completely pure set with only one class) to 1 (maximum impurity in a binary classification).

# Entropy vs Gini Impurity




$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

# Information Gain

.

- Measures the purity of a split; higher information gain is preferred.

$$\text{Information Gain} = \text{Entropy (Parent)} - \sum_{j=1}^{k} \frac{|D_j|}{|D|} \cdot \text{Entropy}(D_j)$$

Information Gain is maximized when a split creates subsets that are as pure as possible.

In decision trees, the attribute with the highest information gain is chosen as the splitting criterion at each node.

# Example.

Consider a dataset of 10 instances, where each instance has two features, “Weather” and “Windy,” and a target variable “Play” (whether to play outside or not).

**Step 1**: Calculate Initial Impurity of the Target Variable “Play”

Total instances: 10

- Yes (Play): 6 instances
- No (Play): 4 instances

| Weather | Windy | Play |
|---|---|---|
| Sunny | No | No |
| Sunny | Yes | No |
| Overcast | No | Yes |
| Rainy | No | Yes |
| Rainy | Yes | No |
| Rainy | No | Yes |
| Overcast | Yes | Yes |
| Sunny | No | No |
| Sunny | No | Yes |
| Rainy | No | Yes |

# Example

- **Using Gini Impurity**

$$\text{Gini} = 1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2$$

= 1 - 0.36 - 0.16 = 0.48

**Using Entropy**

$$\text{Entropy} = -\left(\frac{6}{10}\right) \cdot \log_2\left(\frac{6}{10}\right) - \left(\frac{4}{10}\right) \cdot \log_2\left(\frac{4}{10}\right)$$

$$= -(0.6 \cdot -0.737) - (0.4 \cdot -1.321)$$

= 0.971

Step 2: Calculate Impurity for Splits on “Weather” Feature

Now we calculate the impurity for each possible split.

For “Weather = Sunny”

Sunny subset:  Yes = 1, No = 3

- Gini for Sunny: 1 - (1/4)^2 - (3/4)^2 = 1 - 0.0625 - 0.5625 = 0.375
- Entropy for Sunny:  - (1/4 \cdot \log_2(1/4)) - (3/4 \cdot \log_2(3/4)) = 0.811

For “Weather = Overcast”

**Overcast** subset:

- Yes = 2, No = 0
- Gini for Overcast: 1 - (2/2)^2 = 0 (pure)
- Entropy for Overcast: 0 (pure

# Random Forest Tree

# Introduction

.

Definition: Random Forest is an ensemble learning technique that builds multiple decision trees and merges their results for a more accurate and stable prediction.

Relation to Decision Trees: A Random Forest consists of many Decision Trees, each trained on a random subset of the data and features.

Goal: Improve accuracy and reduce overfitting by combining predictions from multiple trees.

# How Random Forest Works

.

1. **Bootstrapping**: Randomly sample the data with replacement to create multiple training datasets.
2. **Random Feature Selection**: For each tree, select a random subset of features.
3. **Build Decision Trees**: Train each tree independently on the bootstrapped data and subset of features.
4. **Aggregate Results**: For classification, each tree votes, and the majority class is selected. For regression, the average prediction across trees is used.

# Similarities with Decision Trees

.

1. **Base Learner**: Each tree in the forest is a Decision Tree.
2. **Interpretability**: Like Decision Trees, each tree can be visualized for interpretation, although the entire forest may be complex.
3. **Feature Splitting**: Each Decision Tree within the Random Forest performs splits based on features, just as a single Decision Tree does.

# Key Points about Random Forest

.

**Robustness**: Reduces overfitting compared to individual Decision Trees, especially with high-dimensional data.

**Versatility**: Suitable for both classification and regression tasks.

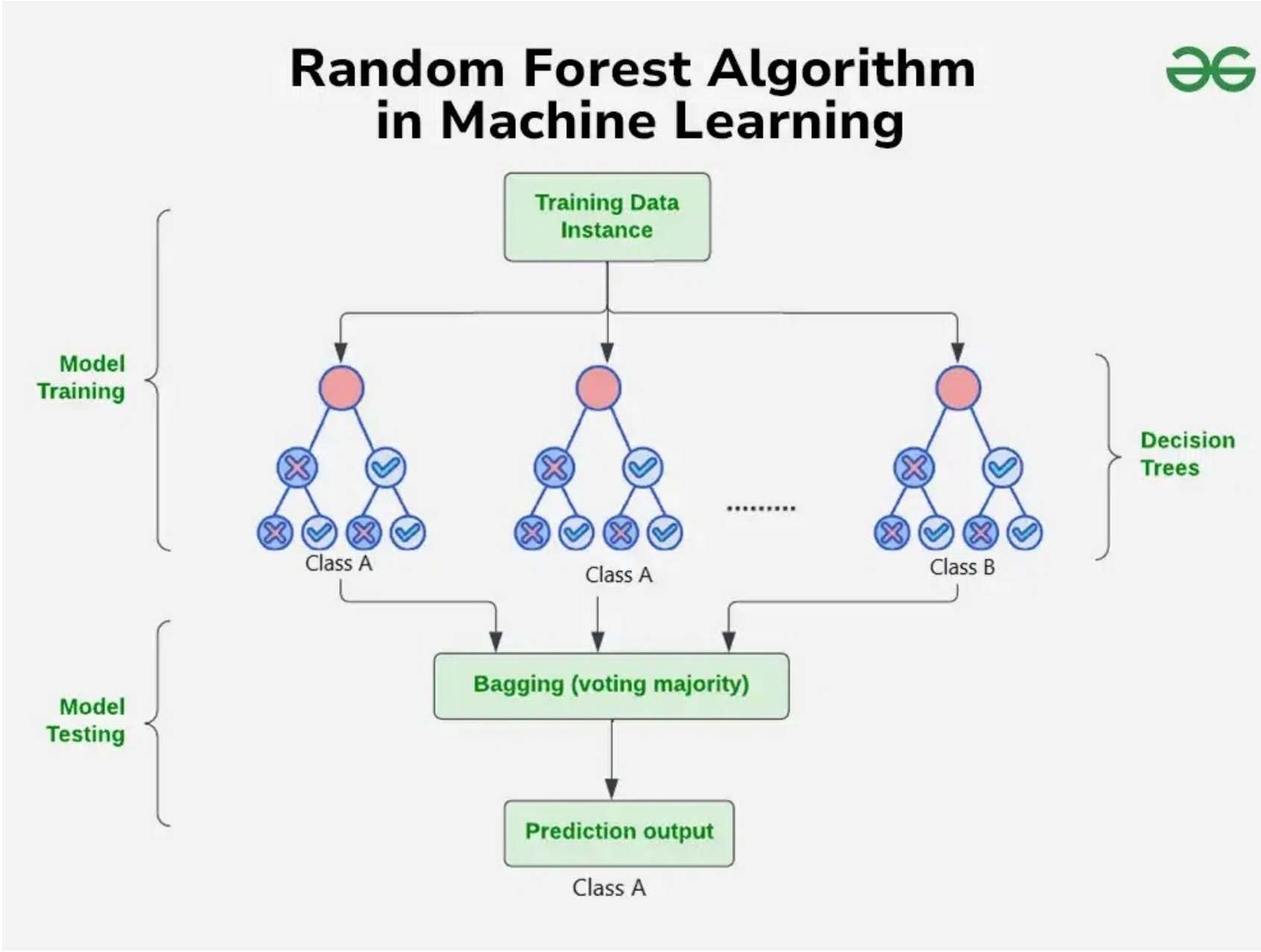
**Scalability**: Effective with large datasets and numerous features.

**Accuracy**: Often achieves high predictive accuracy by aggregating multiple Decision Trees.

**Randomness**: Random sampling (bootstrapping) and feature selection improve model generalization.

# Example

- .


Random Forest Algorithm in Machine Learning
Training Data Instance
Model Training
Class A
Class A
.....
Class B
Decision Trees
Model Testing
Bagging (voting majority)
Prediction output
Class A

.

# Random Forest Algorithm Steps

▪

**Bootstrap Sampling**: Draw multiple random samples from the dataset with replacement.

2. **Tree Construction**: For each bootstrap sample:
   - Select a random subset of features.
   - Build a Decision Tree without pruning.
3. Prediction Aggregation:
   - **For Classification**: Each tree “votes” on the most common class.
   - **For Regression**: Take the average prediction from all trees.
4. **Final Prediction**: Based on the majority vote or average across all trees.

# Application

.

**Healthcare**: Diagnosis prediction and disease classification.

**Finance**: Fraud detection and credit scoring.

**Marketing**: Customer segmentation and churn prediction.

**Environmental Science**: Predicting air quality, forest cover type classification.

# Ensemble Learning

# Introduction

.

**Definition**: Ensemble Learning combines multiple models (learners) to improve overall performance compared to a single model.

**Goal**: Increase predictive accuracy, stability, and reduce errors by leveraging multiple models.

- Types of Ensemble Learning:
- **Bagging** (Bootstrap Aggregating)
- Boosting
- Stacking

# Key concepts

.

**Diversity**: Ensemble models work best when individual learners are diverse, meaning they make different kinds of errors.

**Aggregation**: Combining predictions from multiple models, often by voting (for classification) or averaging (for regression).

**Reduction of Overfitting**: Ensembles help in reducing overfitting, especially when individual models are high variance models (e.g., Decision Trees).

# Bootstrapping

.

**Definition**: Bootstrapping is a resampling technique where samples are drawn randomly with replacement from a dataset to create multiple training datasets.

**Purpose in Ensembles**: Each model in an ensemble trains on different bootstrapped datasets, which improves diversity among models.

Steps:

1. Randomly select data points with replacement to form a “bootstrapped” dataset.

2. Create multiple datasets (often the same size as the original dataset).

3. Train models on these diverse bootstrapped datasets.a

# Bagging (Bootstrap Aggregating)

.

**Definition**: Bagging (Bootstrap Aggregating) is an ensemble method that applies bootstrapping to create multiple models and then aggregates their predictions.

Mechanism:

- Each model is trained on a different bootstrapped dataset.
- The predictions from all models are averaged (regression) or a majority vote is taken (classification).
- **Popular Use**: Bagging is commonly used with high-variance models, like Decision Trees, to reduce overfitting.

# Important Points about Bagging

.

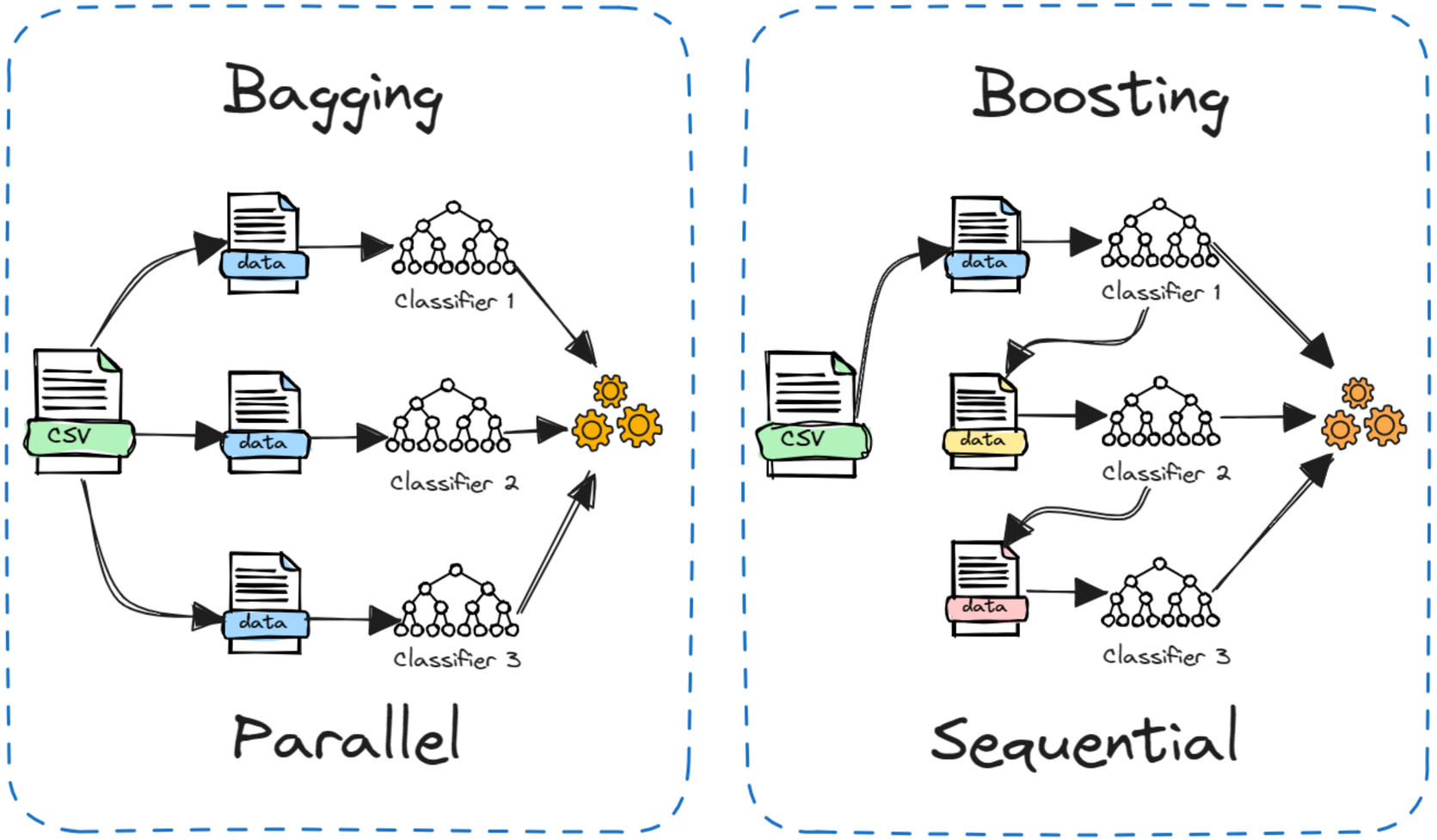
**Randomness**: Using different data samples through bootstrapping introduces randomness and enhances model diversity.

**Accuracy Boost**: Combining predictions typically results in higher accuracy than a single model.

**Overfitting Reduction**: Bagging helps reduce overfitting in high-variance models.

**Popular Algorithm**: Random Forest is one of the most widely known bagging-based ensemble methods.

# Diagram


Bagging
CSV
data
data
data
Classifier 1
Classifier 2
Classifier 3
Parallel
Boosting
CSV
data
data
data
Classifier 1
Classifier 2
Classifier 3
Sequential