MAE (Mean Absolute Error)

📘 What is it?

The **average of the absolute differences** between predicted and actual values.
It tells you **how wrong your predictions are**, on average.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

📌 Example:

Actual: [3, 5, 7]
Predicted: [2, 5, 8]
Errors = [1, 0, 1] → MAE = (1 + 0 + 1) / 3 = **0.67**

✅ Pros:

- Simple to interpret
- Not sensitive to outliers

❌ Cons:

- Does not penalize large errors more than small ones

2. RMSE (Root Mean Squared Error)

📘 What is it?

The square root of the average of **squared** differences.
Gives **more weight to larger errors**.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

📌 Example:

Actual: [3, 5, 7]
Predicted: [2, 5, 8]
Squared Errors = [1, 0, 1] → RMSE = √(2 / 3) ≈ **0.82**

✅ Pros:

- Penalizes large errors more than MAE
- Popular in many applications (e.g., forecasting)

❌ Cons:

- Sensitive to outliers

3. R^2 Score (Coefficient of Determination)

📘 What is it?

Measures the **proportion of variance explained** by the model.
Ranges from **−∞ to 1**. Closer to **1** means better fit.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Where $\bar{y}$ is the mean of actual values.

📌 Example:

If $R^2 = 0.85$ → the model explains **85%** of the variability in the data.

✅ Pros:

- Indicates how well the model explains data
- Interpretable (like 85% of variation is explained)

❌ Cons:

- Can be negative (for poor models)
- Doesn't show **how much** the predictions deviate

When to Use What?

- Use **MAE** when:
  - You want interpretability (e.g., "5 units off on average")
  - Outliers should not dominate
- Use **RMSE** when:
  - You want to **penalize large errors more**
  - More sensitive models (like deep learning, time series)
- Use **R^2** when:
  - You want to **evaluate overall model fit**
  - Used for comparing models on the same dataset

==

Decision Trees into two main categories:

🌳 1. Classification Trees (CART for Classification)

- **Purpose**: Predict **categorical** outcomes (e.g., "Yes" or "No", class 0 or 1)
- **Splitting Criteria**: Uses **Gini impurity** or **Entropy (Information Gain)** to split nodes.
- **Output**: A **class label** (like 0 or 1)

📈 2. Regression Trees (CART for Regression)

- **Purpose**: Predict **continuous** outcomes (e.g., prices, age, temperature)
- **Splitting Criteria**: Uses **Mean Squared Error (MSE)** or **Mean Absolute Error (MAE)**
- **Output**: A **numeric value** (like 45.2)

🔁 Comparison Table

Feature	Classification Tree	Regression Tree
Target Output	Categorical (Discrete Labels)	Continuous (Real-valued numbers)
Splitting Metric	Gini, Entropy	MSE, MAE
Use Cases	Spam detection, Loan approval	Price prediction, Age estimation
Output at Leaf	Most common class	Average of outputs in the leaf

Criterion	Used in	Description
`gini`	Classification	Gini impurity
`entropy`	Classification	Information gain
`squared_error`	Regression	Mean Squared Error (default)
`friedman_mse`	Regression	Faster variant of MSE for boosting
`absolute_error`	Regression	Mean Absolute Error (robust to outliers)
`poisson`	Regression	For count data (Poisson-distributed)

1. Random Forest

✅ Key Features:

- **Type**: Bagging ensemble
- **Base Learner**: Decision Trees
- **Technique**: Combines multiple decision trees trained on bootstrapped samples and averages their predictions (for regression) or uses majority voting (for classification).
- **Key Parameters**:
  - `n_estimators`: Number of trees
  - `max_depth`: Maximum depth of each tree
  - `criterion`: Gini or Entropy (for classification)

🧠 Concept:

Each tree is trained on a different random subset of data, with a random subset of features. This reduces overfitting and increases generalization.

2. Gradient Boosting Machines (GBM)

✅ Key Features:

- **Type**: Boosting ensemble
- **Base Learner**: Typically shallow decision trees
- **Technique**: Sequentially builds trees where each new tree corrects errors from the previous one.
- **Key Parameters**:
  - `n_estimators`: Number of boosting rounds
  - `learning_rate`: Shrinks the contribution of each tree

- `max_depth`: Maximum depth of individual trees

🧠 Concept:

Instead of training all trees independently, GBM builds them sequentially, minimizing a loss function.

3. Model Stacking

✅ Key Features:

- **Type**: Meta-ensemble method
- **Base Learners**: Can use any models (e.g., logistic regression, SVM, decision trees)
- **Technique**: Combines multiple model predictions using a meta-model (e.g., another logistic regression)

🧠 Concept:

Train several base models and then use their outputs as features for a second-level model (meta-learner) to make the final prediction.

Method	Handles Overfitting	Learns from Errors	Works in Parallel	Complexity	Customization
Random Forest	✅	❌ (Independent trees)	✅	Medium	Moderate
Gradient Boosting	⚠️ (Needs tuning)	✅ (Sequential trees)	❌	High	High
Model Stacking	✅	✅ (via meta-model)	⚠️	High	High